BEST AVAILABLE IMAGES

Defective images within this document are accurate representation of The original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

H9
H6-US
120

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification [7]: G06F 17/60 — A2

(11) International Publication Number: **WO 00/39725**

(43) International Publication Date: 6 July 2000 (06.07.00)

(21) International Application Number: PCT/US99/30356

(22) International Filing Date: 21 December 1999 (21.12.99)

(30) Priority Data:
09/220,547 23 December 1998 (23.12.98) US

(71) Applicant: NET PERCEPTIONS, INC. [US/US]; Suite 300, 7901 Flying Cloud Drive, Eden Prairie, MN 55344-7905 (US).

(72) Inventors: BIEGANSKI, Paul; 6461 Regency Lane, Minneapolis, MN 55344 (US). KONSTAN, Joseph, A.; 582 Cretin Avenue South, St. Paul, MN 55116 (US). RAUSER, John; 2720 Aldrich Avenue South, Minneapolis, MN 55408 (US). FRANKOWSKI, Dan; 3216 Colfax Avenue South, Minneapolis, MN 55408 (US).

(74) Agents: GARRETT, Arthur, S. et al.; Finnegan, Henderson, Farabow, Garrett & Dunner, L.L.P., 1300 I Street, N.W., Washington, DC 20005-3315 (US).

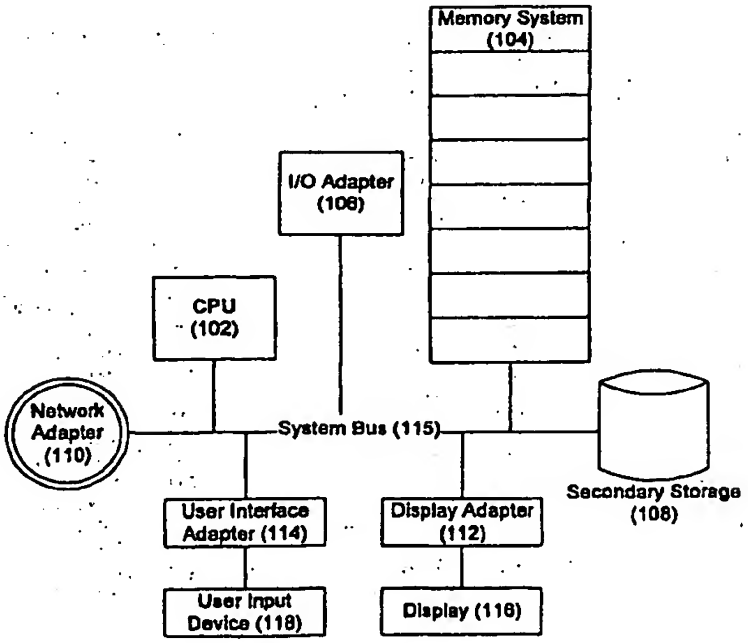
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR PRODUCING ITEM COMPATIBLE RECOMMENDATIONS

(57) Abstract

A recommendation process includes the consideration of the compatibility of the items to be recommended. An electronic processing system for generating a compatibility-modified recommendation output includes a processing system of one or more sets of processors configured to receive applicable data including i) item recommandation data, and ii) item compatibility rules, and to modify the item recommandation data using the item compatibility rules to produce a compatibility-modified recommandation output set. A method of producing a compatibility-modified recommandation includes receiving applicable data, including i) item recommandation data, and ii) item compatibility rules, and modifying the item recommandation data using the item compatibility rules to produce a compatibility-modified recommandation output set.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR PRODUCING ITEM COMPATIBLE RECOMMENDATIONS

Related Applications

This application is related to the following U.S. patent and patent applications, which are incorporated by reference:

1. SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR USING RECEIVER OPERATING CURVES TO EVALUATE PREDICTIVE UTILITY, U.S. Patent No. 5,842,199.

2. SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR UTILIZING IMPLICIT RATINGS IN PREDICTION INFORMATION SYSTEMS, filed October 8, 1996, serial no. 08/725,580.

3. SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR GENERATING IMPLICIT RATINGS BASED ON RECEIVER OPERATING CURVES, filed October 8, 1996, serial no. 08/729,787.

4. SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR INCREASING THE USER VALUE OF RECOMMENDATIONS MADE BY A RECOMMENDER SYSTEM, filed July 17, 1998, by P. Bieganski, application serial no. 09/118,025.

5. SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR MAKING HIGH USER VALUE RECOMMENDATIONS, filed July 17, 1998, by P. Bieganski, application serial no. 09/118,026.

Background

The present invention relates generally to data processing systems, and more particularly, collaborative filtering and recommender systems.

Recommender systems predict the preferences of users based on attributes known about the user or a past history of preferences or consumption by the user. For example, a recommender system may predict that a user will like the movie "Titanic" because he previously indicated a liking for such other epic movies as "Lawrence of Arabia" or "Ben Hur".

Present recommenders focus on making accurate recommendations of user preference. However, the most accurate recommendations, in isolation, may be

the wrong items when considered collectively as a set of recommendations and as recommendations designed to accompany products that the user is buying or has bought. For example, a system recommending books may discover that a particular customer would enjoy Tolkien's *The Hobbit*, based on other books that the user did and did not like. *The Hobbit*, however, appears in more than thirty editions, from hardcover to paperback, audiobook to videocassette, editions bundled with other books, and editions illustrated by different illustrators. If the recommender system were to suggest that the customer consider several of these, it would likely annoy the user and waste the opportunity to recommend a more diverse set of books. As important, if we know that the customer already owns one edition of *The Hobbit*, we likely would prefer to avoid recommending another edition, so as to add greater value to the bookstore's recommendations. Similar examples also occur where having an item makes recommending a different item more valuable. For example, a student who has a copy of a particular textbook in her shopping cart would be well-served by a recommendation for the study guide that accompanies that textbook, even if her prior purchases do not independently indicate that the study guide is an item she would otherwise be likely to buy.

A recommender system determines its recommendations by examining previous user preference data. The preference data can be unary or numerically valued. Unary preference data is a set of customer-item pairs: a customer-item pair indicates that an event linking the customer to the item has occurred. No additional preference information is available to the recommender system about the a user-item event except that it happened. The non-existence of a customer-item pair (more generally known as a tuple) for a specific customer-item pair does not indicate a preference: it only indicates a lack of information. An example of unary customer data is purchase record data where a customer-item pair indicates that the customer has purchased the indicated item. Another example of unary data is contained in web page logs, where a customer-item pair indicates that the customer has visited a specific web page.

Binary and numerically valued preference data are generally in the form of a 3-tuples, where the three elements of the tuple are customer identifier, item

identifier, and preference value. The preference value indicates, for example, the strength of the user's preference for the item or whether the user's preference is either for or against the item. To illustrate, where the preference is represented in binary form, a "0" may mean a preference against an item while a "1" means a preference for the item. Where the preference is presented as numerically valued data, the data value may represent a one-dimensional axis of preference, with the midpoint indicating an ambivalent preference for the item, a low value indicating a strong dislike for the item, and a high value indicating a strong preference for the item.

Preference data may be presented to the recommender system in explicit or implicit form. Explicit preference data are preference values that a user has supplied directly, for example by filling out a survey. Implicit preference data consist of preference values that have been inferred by observing actions that the user has taken. It can be inferred that the user has some preference for the item that she has just bought, although the act of purchasing the item is not an explicit statement of preference *per se*. A user's preference for a web page may be inferred, for example, by measuring the amount of time that the user spends reading the web page, or the number of times the user returns to that page.

The inputs to a recommender system are typically preference values as described above. The outputs of the recommender system are predictions of preference values for items, particularly those for which the user has not already indicated a preference. Like the input values, the output preferences may be unary, binary, or numerically valued. A system that outputs unary recommendations predicts items that will be of interest to the user, but does not attempt to predict the strength of a user's preference for each item. Binary predictions indicate items that are likely to be of high preference to the user and items that are likely to be of low preference, but again cannot provide an estimate of preference strength. Numerically valued preferences indicate a preference for or against the item and also indicate the preference strength. Note that the domain of the preference input may be different from the domain of the output preference predictions. For example, the preference input may be unary, while the output preference predictions may be numerically valued.

While unary and binary preference values do not indicate the strength of the preference, some recommender systems may additionally rank the preference predictions being returned such that the highest rank predictions have the largest probability of being correct. Numerically valued items are implicitly ranked.

Existing recommender systems generate recommendations by selecting the highest-ranking positive preference values. However, this technique does not always provide a desirable effect. In some cases, there may be strong correlative effects between a current or past purchase, and a recommendation that otherwise would have a low ranking. For example, the purchase of 35 mm film may rank low on a recommendation list, or may not even be on the recommendation list, given the current contents of a shopping basket. However, if there is knowledge in the recommender system that the user has previously purchased a 35 mm camera, then it becomes more sensible to recommend 35 mm film to the user.

There may also be strong anti-correlative effects that should be taken into account to remove a recommendation from a recommendation list. For example, if a user has purchased a pizza and a bottle of root beer made by one manufacturer, then, there is little value in recommending that the user by root beer made by another manufacturer.

Therefore, there exists a problem with existing recommender systems that, although able to recommend items with high confidence level, the recommender system is unable determine the quality of recommendations in view of other items present in the shopping basket, the recommendation set or in an historical set of past purchases. Consequently, the value of some of the recommendations made to a user may be low. There is a need to reduce the frequency of occurrence of low value recommendations.

Summary of the Invention

To address the problems listed above, the present invention is directed to including compatibility in the process of recommending items to a user. In particular, the invention is directed to an electronic processing system for generating a

compatibility-modified recommendation output set to a user based, at least in part, on a set of item compatibility rules. The system includes a processing system of one or more sets of processors configured to receive applicable data including i) item recommendation data, and ii) item compatibility rules, and to modify the item recommendation data using the item compatibility rules to produce a compatibility-modified recommendation output set.

In another embodiment, the invention is directed to a method of producing a compatibility filtered and weighted recommendation to a user, using a computer having a processing system with one or more sets of processors. The method includes receiving, using the processing system, applicable data, including i) item recommendation data, and ii) item compatibility rules, and modifying the item recommendation data, with the processing system, using the item compatibility rules to produce a compatibility-modified recommendation output set.

In another embodiment, the invention is directed to a computer-readable program storage device, having a set of program instructions physically embodied thereon, executable by a computer, to perform a method of producing a compatibility filtered and weighted recommendation. The method includes receiving applicable data, including i) item recommendation data, and ii) item compatibility rules, and modifying the item recommendation data using the item compatibility rules to produce a compatibility-modified recommendation output set.

The above summary of the present invention is not intended to describe each illustrated embodiment or every implementation of the present invention. Other features of the invention, together with a fuller understanding of the invention will become apparent and appreciated by referring to the following description and claims taken in conjunction with the accompanying drawings.

Brief Description of the Drawings

The invention may be more completely understood in consideration of the following detailed description of various embodiments of the invention in connection with the accompanying drawings, in which:

FIG. 1 illustrates a computer system for use with an embodiment of the present invention;

FIG. 2 illustrates a system for generating a compatibility-modified recommendation output set for a user according to one embodiment of the invention;

FIG. 3 illustrates a method of implementing a compatibility modifier;

FIG. 4 illustrates a method of modifying a set of recommendations to produce a compatibility-modified recommendation output set;

FIG. 5 illustrates methods for modifying items in a recommendation set with a recommendation score in a compatibility modifier;

FIG. 6 illustrates a compatibility-aware recommendation system according to another embodiment of the invention;

FIG. 7 illustrates a method of implementing compatibility awareness within a recommendation engine;

FIG. 8 illustrates methods for modifying items in a recommendation set with a recommendation score in a compatibility aware recommendation engine;

FIG. 9 illustrates data structures and methods for inferring item compatibility rules from purchase data;

FIG. 10 illustrates two call center information systems that receive item recommendations, one system being compatibility-unaware and the other including a compatibility filter;

FIG. 11 illustrates an information system for a commerce center for performing electronic transactions;

FIG. 12 illustrates an information system for a telephone call center;

FIG. 13 illustrates an electronic system for use with a cash register; and

FIG. 14A and 14B illustrate different processor configurations for a compatibility modifier.

While the invention is amenable to various modifications and alternative forms, specifics thereof have been shown by way of example in the drawings and will be described in detail. It should be understood, however, that the intention is not to limit the invention to the particular embodiments described. On the contrary, the

intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the invention as defined by the appended claims.

Detailed Description

This invention is applicable to recommender systems, and is believed to be particularly applicable to increasing the value of recommendations made by recommender systems. In particular, the invention is believed to be applicable to increasing the value of recommendations made by existing recommender systems passing the recommendations through a compatibility modifier to add items to, or subtract items from, the recommendation list.

Recommender systems concentrate on making accurate recommendations, but do not take account of the compatibility of a recommendation with other items that may be in the shopping basket, in historical purchase data or in the present recommendation set. Recommender systems, therefore, would benefit from incorporating *item compatibility*. We define item compatibility as the relationships among items that make some items *substitutes* or *complements* for other items.

A substitute relationship exists when a customer generally consumes one item or the other, but not both. Examples of substitutes include skim milk and whole milk. Most customers do not purchase these items together. We should note that for compatibility purposes, substitutes need not actually replace each other in a precise way. For example, tofu and bacon may be substitutes for each other if the people who consume tofu generally do not consume bacon or vice versa. In the book example, each edition of the Hobbit is a substitute for other editions, though the strength of the relationship may vary. For example, someone who owns a complete set of Tolkien novels is unlikely to buy a separate copy of *The Hobbit*, but someone who owns only a copy of *The Hobbit* may be willing to buy the complete set, since it includes many other novels.

A complement relationship exists when the fact that a customer owns or consumes one item generally indicates that the customer will prefer or consume another item or set of items. For example, sugar and cream are both complements for coffee, and hot dog buns and mustard are complements for hot dogs. Complements may be either one–way or two–way. For example, the purchase of eggs may not indicate a

likely purchase of cake mix, but the purchase of cake mix may strongly suggest the purchase of eggs, since eggs are used with cake mix to make a cake. Other examples of complements include film for cameras, batteries for battery–operated toys and electronics, and video cassettes for VCRs. Note that some complement relationships persist for a long period after the product is purchased (e.g., owning a camera leads to a long sequence of film purchases), while others are more immediate (e.g., having bought cake mix in the past doesn't generally increase the future likelihood of buying eggs. In the textbook example provided above, a study guide is a likely complement for an associated textbook, and may have a value that persists for a period of time corresponding to the duration of the course of study.

A recommendation set is defined as a set of items recommended to a user. The recommendation set may be unordered, in which case the weight accorded to each recommended item is equal. The recommendation set may be ordered, in which case some items (generally those at the top of the recommendation list) are recommended more strongly than others. The recommendation set may include recommendation values associated with each item, in which case the strength with which each item is recommended is indicated by the associated value.

Recommendation sets may be generated in many ways. Common mechanisms include "word of mouth" wherein a person suggests items to another person, publication of individual opinions such as movie or restaurant reviews written by critics (which often have stars to indicate a recommendation value); publication of collected opinions such as the automobile reliability and movie ratings published by *Consumer Reports*; evaluation by systematic and possibly objective ratings formulas as is commonly done for comparative product reviews; evaluation by software systems that evaluate the contents of the items being considered as is commonly done for information retrieval searches such as library and world wide web searchers; and collaborative filtering systems that use the opinions of multiple users to create a recommendation for other users. An interesting case is the use of a recommendation engine, which incorporates collaborative filtering, content analysis, or both to automate the process of generating recommendations. The present invention is applicable to recommendation sets generated in all manners, including those recommendation sets generated by recommendation engines.

Compatibility–modified recommendation sets are recommendation sets that incorporate knowledge about item compatibility and knowledge about the items that are already being recommended to, being purchased by, or previously purchased by the user. A compatibility–modified recommendation set may exclude items that would be recommended without compatibility information; it may recommend items that would not have been recommended without compatibility information; and it may reorder or assign new recommendation values to items being recommended.

Compatibility–modified recommendation sets provide greater value to the users of a recommendation system. The customer receiving the recommendations receives a set of recommendations that is less likely to recommend items that conflict with other items and is more likely to contain items compatible with those items being purchased, either presently or previously, or being recommended.

A marketer who uses compatibility–modified recommendations is provided with a set of recommendations that is more likely to anticipate real customer interests, and therefore is more likely to lead to successful suggestive selling.

The present invention is directed to the creation of compatibility–modified recommendation sets. It is useful for improving the quality of a recommendation set by taking advantage of knowledge of item compatibility, in the form of item compatibility rules, and taking advantage of the knowledge of which items a customer has previously purchased, is presently purchasing, or are being presently recommended to a user.

The present invention may be used to modify a recommendation set that is provided from any source, or may be employed in a recommendation engine that uses compatibility rules to generate compatibility–modified recommendations directly.

It is important to note that, although much of the description contained herein refers to implementations that use a particular form of item compatibility rule, namely one–way item–to–item substitution and complement rules, the invention is not limited to this form of rule. The invention described herein can apply item compatibility rules in any form to recommendations. It is also important to note that, although much of the description contained herein refers to unordered recommendation sets, the invention is not limited to unordered recommendation sets and may be used to modify recommendation sets of any type to improve their value by incorporating knowledge about item compatibility.

FIG. 1 shows a diagram of a typical computer system suitable for practicing the present invention. The computer may include a system of one or more central processing units (CPUs) 102, a memory system 104, an input/output (I/O) adapter 106, a secondary storage 108, a network interface 110, a user interface adapter 114, and a display adapter 112. All of the computer components are connected by a system bus 115. The display adapter 112 may be connected to a display 116 for displaying a recommendation to a user. The user interface adapter 114 may be connected to a user input device 118, such as a keyboard, mouse, barcode scanner or the like.

The computer system may include more than one processor, where the processors are in different locations. In such a case, the processors may be linked by input/output interfaces over a network, such as a local area network, wide area network or the Internet.

In one particular embodiment of the invention, illustrated in FIG. 2, a compatibility modified recommendation set 205 is generated by a compatibility modifier 200 that uses a recommendation set 201 and a set of item compatibility rules 204. The compatibility modifier may also use a shopping set 202 and an historical set 203. The generation of the compatibility modified recommendation set 205 in this embodiment requires the use of the recommendation set 201 and the item compatibility rules 204. The shopping set 202 and the history set 205 are optional and may be used in any combination with the recommendation set 201 and the item compatibility rules 204. The item compatibility rules 204 are used to determine which items should be added to, removed from, and changed in the recommendation set 201 to yield the compatibility modified recommendation set 205.

There are many different ways to determine recommendation sets 201, shopping sets 202, history sets 203, and item compatibility rules 204. Each of the sets and rules is based on a universal set of *items* that defines the products, services, or other goods that customers may buy or own. The universal set of *items* is extremely flexible, and is generally customized for each application. For some retail applications, the universal item set exactly matches the products for sale by a particular retailer. In other retail applications, the universal item set is augmented to include products that a user may have purchased elsewhere that are not sold by the retailer, combinations of products that form common sets, or product aggregates and category items such as

"Tolkien" for a bookstore or "produce" for a grocery store. Other applications include items that correspond to demographic or psychographic properties that a customer may possess. The present invention works with all such item sets. For purposes of clarity, we refer here to the subset of items that may be recommended (usually the same as or a subset of those for sale) as the *recommendable items*.

A recommendation set 201 is a set of recommendations on the recommendable items. These recommendations may be provided by an outside source or generated automatically using, for example, genetic algorithms, collaborative filtering, neural networks, or other statistical models. The recommendations may also be pre–determined based on vendor specifications, or derived from human or computer–based experts. For example, the recommendation set in a book store may be a list of recommended books that employees and book reviewers found interesting. In a travel agency, the recommendation set may consist of hotels and restaurants contained in the travel agency's travel guide. For specific applications where shopping sets and/or historical sets are used, and for specific users at specific times, the recommendation set may be empty; an empty recommendation set simply indicates that the system has no recommendable items to suggest at that time. An empty set can still be compatibility–modified, since the modification may add new items to the set.

A shopping set 202 includes items for which the user has indicated a current intent to purchase or consume. The user's indication may be explicitly declared or inferred from user actions. Examples of ways that a shopping set can be generated include, but are not limited to: using the contents of a "market basket" from an Internet–based shopping service; using an active shopping cart that scans the bar codes of products being placed in the basket; using the set of items entered into a cash register at check–out time; and using a shopping list provided by the customer. A shopping set may also be generated by observing the customer's behavior. It is advantageous to use a shopping set when the value of recommendations provided to the user can be increased based on their compatibility with products currently being purchased. Hence, a cooking store might use the shopping set at check–out to identify items for suggestive selling, such as a Chinese cookbook for customers purchasing a Wok.

An historical set 203 consists of explicit or implicit information about the user's possession of and preference for items. The historical set may include a list of items selected or purchased by the user in past interactions with the system, or just

purchased recently; a list of items that the user owns, whether purchased through the system or elsewhere; a list of user ratings of items, based upon expressed user preferences, inferred preferences, or other data. The historical set may also include any other set of data that represents information about the user's possession of and preference for items before the present interaction with the system. Historical data may, for example, include data gathered from credit card records, marketing surveys, and other commercially available sources of individualized preference data. For example, a store that sells film and batteries may use the historical set to store the types of cameras owned by the user, since they help predict which film and batteries to recommend. A grocery store may record the type of milk purchases, since a customer who regularly purchases whole milk may not value a recommendation for skim milk. A bookstore may record a list of the books purchased by the customer to allow it to recommend sequels to books already purchased and to avoid recommending books that are incompatible with the ones already owned.

The terms "elements", "sets" and "collections" are used in the description below. An element is typically a specific product that can be selected by the customer, whereas a set is generally a less specific grouping of elements. For example, a 32 oz. jar of "Hellman's™ mayonnaise" and a 32 oz. jar of "Kraft™ mayonnaise" are examples of elements, while "mayonnaise" describes a set of different elements that includes "Hellman's™ mayonnaise" and "Kraft™ mayonnaise". A collection is a grouping of elements or sets of elements that may be associated with each other. For example, mayonnaise and dijon mustard may be placed in a collection because the marketer wishes to express the knowledge that a customer buying both mayonnaise and dijon mustard would be likely or unlikely to purchase "Dijonaise"™, which is a mixture of both mayonnaise and dijon mustard. The term "item" is used as a generic term to describe elements, sets and collections.

Item compatibility rules 204 express compatibility relationships among items. There are many ways of representing item compatibility: unidirectional implication rules, bidirectional implication rules, and generalized rules of various types, including multi–way rules. The rules themselves may relate individual items, sets of items, or specific collections, and they may include weight factors or priorities to indicate the relative importance of or accuracy of the rule. The rules may also be

unweighted an unprioritized. Generally, the easiest rules to use are unweighted unidirectional rules that relate one item to another. Such rules are expressed in the following manner:

Complement rule:	**A–> B**	**A *implies* B**
Substitute rule:	**C –> ~D**	**C *implies* not D**

The basic complement rule means: when item A is recommended to, purchased by, or owned by a user, then item B is likely to be a good companion item. For example, if A is "Diet Cereal" and B is "Skim Milk" then the relationship A –> B indicates that people who buy "Diet Cereal" are also likely to be interested in "Skim Milk."

The basic substitute rule means: when item C is recommended to, purchased by, or owned by a user, then item D is unlikely to be a good companion item. For example, if C is "Manufacturer A's mayonnaise" and D is "Manufacturer B's mayonnaise" then the relationship C –> ~D indicates that people are unlikely to buy mayonnaise from both Manufacturer A and Manufacturer B.

Bidirectional implication rules specify two–way relationships. For example, the bidirectional complement rule A <–> B indicates that A is a complement of B and B is a complement of A. This is a more general, and less powerful mechanism than unidirectional rules, since every rule X <–> Y can be transformed into two rules: X –> Y and Y –> X. However, there is no bidirectional way to represent that most "Diet Cereal" customers buy skim milk, but relatively few skim milk customers buy "Diet Cereal". Similarly, bidirectional substitute rules may be transformed into a pair of unidirectional rules with no loss of information.

More generalized forms of expressing rules exist, including rules that have multiple implications. These rules may be used in the present invention, either directly by the compatibility modifier 200 or by transforming the rules into a form directly usable by the compatibility modifier 200.

In addition to relating individual items, rules may also relate sets or collections of items. A rule relates a set of items when the left–hand–side (input), the right–hand–side (output), or both sides (input and output) contain an explicit or implied list of items, any one of which can be substituted into the rule. For example, in the rule:

hot–dog–buns –> hot–dogs

hot–dog–buns and **hot–dogs** are both sets that represent all of the different individual products in the category. The meaning of the rule is that the purchase (recommendation) of any item in the **hot–dog–buns** set would be complemented by a recommendation of any product from the **hot–dogs** set. This type of rule can be transformed into a set of unidirectional rules among individual items by creating a rule for each pairing of an item from the left–hand–side (input) set with an item from the right–hand–side (output) set. The resulting number of rules is equal to the product of the number of items in each set.

Similarly, substitute rules may be defined among sets of items. The rule:

hot–dogs –> ~hot–dogs

is a typical rule. It should be interpreted to read that the purchase of any item in the **hot–dogs** set substitutes for (i.e., is incompatible with) recommending any other item in the set. In other words, don't recommend hot dogs to someone who already is buying or receiving a recommendation for other hot dogs.

Item set substitute rules, as well as bi–directional item set compatibility rules can be transformed into item–to–item rules by pairing each item from the left–hand–side set with every item in the right–hand–side set, as suggested above.

Item compatibility rules may include associated weights or values to guide the compatibility modifier 200. The specific weights and values used depend on the particular application. One example involves assigning a rating–scale difference value to each rule. This value corresponds to the amount by which a recommendation should be changed when a rule applies. For example, if **hot–dog–buns –> hot–dogs** with a weight of +2.0, then items in the set **hot–dogs** gain 2 points on the recommendation scale when **hot–dog–buns** items are recommended or purchased.

In a second example, a priority level is assigned to each rule, to allow rules of higher precedence to take priority over ones of lower precedence. For example, a "default" rule of lower priority might indicate that different types of film are substitutes for one another (**film –> ~film**). Higher priority rules may indicate certain types of film as complementary with certain cameras (e.g., **35mm–camera –> 35mm–film** and **110–camera –> 110 film**). The differences in priority may help the compatibility modifier recognize that a customer who owns two different types of

camera may want to buy two different types of film, even though most customers buy only one type.

Item compatibility rules may be created in many different ways. These include, but are not limited to, the following. An individual, such as a marketer, may create the rules and enter them into the system using an input device 118: this may be termed a marketer specification. The rules may be generated automatically by a process external to the present invention, including but not limited, to machine learning and statistical analysis processes such as genetic algorithms, neural networks, and rule inference systems, data mining processes, and other statistical analyses of historical product preference and purchase data. The rules may be stored in computer memory 104 or on a secondary storage device 108 and introduced into the system. Rules may also be created by customers themselves, again entered through an input device 118: this may be termed a customer specification. In an embodiment described below, rules may be inferred within a recommendation engine itself. The rules used by the compatibility modifier 200 may be generated by more than one of these processes.

The compatibility modifier 200 accepts as inputs the recommendation set 201, the compatibility rules 204, and, optionally, the shopping set 202 and/or the history set 203. The compatibility modifier 200 applies the compatibility rules 204 to the recommendation set 201, optionally using the shopping and historical sets 202 and 203, to produce a modified recommendation set . In this embodiment of the invention, the compatibility modifier 200 executes as a process on the computer system, for example computer system as shown in FIG. 1, on one or more processors. The compatibility modifier 200 implements a *modification algorithm*, which is an algorithmic process that applies the rules and generates the modified recommendation set.

An example of a general flow chart to implement a compatibility modifier 200 is illustrated in FIG. 3. The recommendation set 201 is received at step 301. Next, it is determined whether a shopping set 202 exists or not, at step 302. If the shopping set 202 does exist, then it is received, at step 303. Next, it is determined whether a history set 203 exists or not, at step 304. If the history set 203 exists, then it is received by the compatibility modifier 200, at step 305. After all the input data sets 201, 202 and 203 are received, the compatibility modifier 200 receives the item compatibility rules 204, at step 306. **does the compatibility modifier fetch only the rules applicable to the items in the input data sets?** The compatibility modifier 200

uses the rules 204 and the input data sets 202 and 203 to modify the recommendation set 201, at step 307, and outputs compatibility-modified recommendations 205, at step 308.

Several different modification algorithms may be used in the compatibility modifier 200, each of which accomplishes the goal of modifying a recommendation set by applying the compatibility rules to improve the value of the set of recommendations collectively. Some of these algorithms also use information about items currently being considered by the user (i.e., the selection set) and items that have historically been preferred or purchased by the user (i.e., the historical set).

One method for applying unweighted unidirectional rules to an ordered recommendation set is shown in FIGs. 4A and 4B. A generalized form of the method 400 is illustrated in FIG. 4A, while a more specific form is illustrated in FIG. 4B. The following description relates the specific steps shown in FIG. 4B to the general steps of FIG. 4A. In this method 400, a modified recommendation set 208 is produced using the following steps:

1. If a shopping set 202 exists, as determined at step 452, all items that complement items in the shopping set 202 are added to a new, or empty, modified recommendation set. This is a general description of steps 403–408.
2. All items in the original recommendation set 201 are added, at step 454, to the modified recommendation set. This is a generalization of step 409.
3. All items that complement items in the historical set 203 (if any) and the original recommendation set 201 are added, at step 456, to the modified recommendation set. This step is a generalized description of steps 410–416 and steps 417–422.
4. All items that are substitutes for items in the shopping set 202 are removed, at step 458, from the modified recommendation set. This is a generalized description of steps 423–429.
5. Any items that are substitutes for earlier items in the modified recommendation set are removed at step 460. This is a generalized description of steps 430–438.

In following the above steps, the recommendations presented to the user do not include a recommendation not to buy an item present on the shopping list.

Turning now to the specific embodiment illustrated in FIG. 4B, it is important to note that all sets are assumed to be ordered sets without replication. The

UNION operator is assumed to preserve order. Various working sets, **NewSet**, **OutSet**, and **SSet** are initialized at step 401, and a determination is made, at step 402, whether a shopping set 202 exists. If there is no shopping set 202, then the method passes directly to step 409. If there is a shopping set, then the set **SSet** is tested to determine whether it is empty. If it is empty, then the method passes to step 409. If **SSet** is not empty, then the first item in **SSet** is selected out as **Sitem** and a temporary copy of rules **Temp** is initialized, in step 404. After **Temp** is tested for being empty, at step 405, the first rule in **Temp** is extracted as **Rule**, at step 406. If **Rule** is a complement rule and **Sitem** is an input to, in other words on the left hand side of, the **Rule**, as determined at step 407, then the item on the output of , i.e. on the right hand side of, the **Rule** is added to **NewSet**, at step 408. Once all the rules have been tested, as determined at step 405, the method returns to step 403 for the next item in **SSet**.

Once all the complements for the items in the shopping set have been identified and added to **NewSet**, **NewSet** is updated by the addition of the items in the recommendation set **RecSet** using the UNION operator, at step 409.

A determination is made, at step 410, whether the history set **HistorySet** exists. If **HistorySet** does not exist, then the method proceeds to step 417. Where **HistorySet** is determined to exist, then a determination is made whether **HistorySet** is empty, at step 411. If **HistorySet** is empty, then the method proceeds to step 417. Where **HistorySet** is not empty, the first item in **HistorySet** is extracted as **Hitem**, and a temporary copy of rules **Temp** is initialized, in step 412. After **Temp** is tested for being empty, at step 413, the first rule in **Temp** is extracted as **Rule**, at step 414. If **Rule** is a complement rule and **Hitem** is on the left hand side of the **Rule**, as determined at step 415, then the item on the right hand side of **Rule** is added to **NewSet**, at step 416. Once all the rules have been tested, as determined at step 413, the method returns to step 411 for the next item in **HistorySet**.

Once all the complements for the items in the history set have been identified and added to **NewSet**, the complements to the items in the recommendation set are added. This starts first by a determination of whether the recommendation set **RecSet** is empty, at step 417. If **RecSet** is empty, then the method proceeds to step 423. Where **RecSet** is not empty, the first item in **RecSet** is extracted as **Ritem**, and a temporary copy of rules **Temp** is initialized, in step 418. After **Temp** is tested for being

empty, at step 419, the first rule in **Temp** is extracted as **Rule**, at step 420. If **Rule** is a complement rule and **Ritem** is on the left hand side of the **Rule**, as determined at step 421, then the item on the right hand side of **Rule** is added to **NewSet**, at step 422. Once all the rules have been tested, as determined at step 419, the method returns to step 417 for the next item in **RSet**.

Once no more items exist in **RSet**, as determined at step 417, **NewSet** is tested to remove items that are substitutes for items in the shopping set. This commences by determining, at step 424, whether the shopping set **ShoppingSet** is empty. If it is empty, them the method proceeds to step 430. If it is not empty, then the first item in **ShoppingSet** is extracted as **Sitem**, and a temporary copy of rules **Temp** is initialized, in step 425. After **Temp** is tested for being empty, at step 426, the first rule in **Temp** is extracted as **Rule**, at step 427. If **Rule** is a substitute rule and **Sitem** is on the left hand side of the **Rule**, as determined at step 428, then the item on the right hand side of **Rule** is subtracted from **NewSet**, at step 429. Once all the rules have been tested, as determined at step 426, the method returns to step 424 for the next item in **ShoppingSet**.

Once substitutes for all items in **ShoppingSet** have been removed from **NewSet**, the method proceeds to step 430, where a determination is made of whether **NewSet** is empty. If **NewSet** is empty, then the method ends at step 439 by outputting **OutSet**. If **NewSet** is not empty, then **Nitem** is extracted from **NewSet**, **TRule** is initialized as a temporary copy of **RuleSet**, and **TRec** is initialized as a temporary copy of **OutSet**, at step 431. If **TRule** is empty, as determined at step 432, the method proceeds to step 438. If **TRule** is not empty, then the first rule in **TRule** is extracted as **Rule**, at step 433. A determination is made, at step 434, whether **Rule** is a substitute rule and **Nitem** is on the right hand side of **Rule**. If not, then the method returns to step 432. If so, then a determination is made whether **TRec** is empty, at step 435. If **TRec** is empty, then the method returns to step 432. If **TRec** is not empty, then the first item in **TRec** is extracted as **Rec**, at step 436 and a determination made, at step 437, whether **Rec** is on the left hand side of **Rule**. If so, then the method returns to step 430. If not, then the method returns to step 435.

Once **TRule** is determined to be empty, at step 432, **OutSet** is updated by adding **Nitem**, at step 438, after which the method returns to step 430.

This method 400 is useful for applications such as grocery shopping, where it is important to recommend item sets that are compatible with the items currently being purchased by the customer. This is ensured by making it a top priority to place complements to shopping set items at the top of the recommendation set and by removing any items that substitute for items in the shopping set. The second priority is to leverage the recommendations to create a better shopping set. The third priority is to take advantage of historical data. For a shopping application, historical data might include information learned about the customer's pantry behaviors. For example, we may know that a particular customer tends to buy large quantities of pancake mix and therefore trigger a rule that may recommend pancake syrup. An interesting property of this algorithm is that it can function properly even when the recommendation set for a particular user is empty, if that user has a shopping or historical set. In other words, if nothing is known about the customer, no recommendations can be made based on the customer's tastes and historical purchases. However, items can still be suggested that that complement the customer's current selections without substituting for any of those selections.

Another method may be used that operates on a recommendation set in which numerical recommendation scores are associated with the recommendations by applying a rule set that includes numerical difference values with the rules. This second method 500, shown in FIG. 5, constructs a modified recommendation set by applying the rules so as to modify the numerical scores for the recommended items and so as to add items when appropriate. The method 500 does not remove items from the recommendation set; instead it reduces the numerical recommendation score of less favored items. In actual use, the application may filter out recommendations with recommendation scores lower than a certain threshold, or may use low-scoring recommendations to create a "dis-recommended" list, that is a list of items to be avoided after item compatibility has been factored in. This method 500 uses an operation labeled "Apply a Rule" 510 which does the following:

1. If the item on the right-hand-side of the rule is not in the modified recommendation set, that item is added to the modified recommendation set with a recommendation score that represents neutral preference (e.g., 3 on a scale of 1 through 5).

2. For all items on the right-hand-side of a rule, either originally present in the modified recommendation or added in step 1, the rule's difference value is added to (complement rules), or subtracted from (substitute rules) the recommendation score for that item on the right-hand-side.

Another way of performing step 2 is to add a modifier to the recommendation score, where the modifier is positive for a complement rule and negative for a substitute rule.

The remainder 520 of the method is used to create a modified recommendation set as follows:

1. Copy the original recommendation set to initialize the modified recommendation set.
2. Step through each item in the shopping set and historical set, if any. If the item has not already been considered, then apply each rule that has that item on the left-hand-side.
3. Sort the modified recommendation set by numerical recommendation score, such that the first item has the highest score and the last item has the lowest score.
4. Step through each item in the modified recommendation set. If the selected item has a recommendation score below the recommendation threshold, skip it. For each item not skipped, select the set of rules that have the selected item on the left-hand-side. For each rule where the item on the right-hand-side has not already appeared earlier in the modified recommendation set, apply the rule. After applying the rules that apply to the selected item, re-sort the remaining items in the modified recommendation set.
5. Sort the entire modified recommendation set to produce a sorted output set.

This method 500 is advantageous in applications where every item has a recommendation score or where items may intentionally have both high and low recommendation scores. By limiting the application of rules to items being recommended favorably, or to items selected by the user, the method 500 algorithm avoids false recommendations based on compatibility with undesired items. The method 500 may be efficiently implemented using a known set representation, for example one that includes a hash table for rapid lookup and a heap for incremental sorting.

A second embodiment of the invention is shown in FIG. 6. In this embodiment, the compatibility modifier 611 is integrated into a recommendation engine 600. The recommendation engine may be constructed using one of a number of different methods, for example as is disclosed in Communications of the ACM (Association for Computing Machinery) , vol. 40(3), March 1997, in articles by Balabonovic et al., "Content–Based Collaborative Recommendation", pp. 66–72; Kautz et al., "Referral web: Combining social networks and collaborative filtering", pp. 63–65; Konstan et al., "Grouplens: Applying collaborative filtering to Usenet News", pp. 77–87; Resnick et al., "Recommender systems", pp. 56–58; Rucker et al., "Siteseer: Personalized navigation for the web", pp. 73–76; and Terveen et al., "Phoaks: A system for sharing recommendations", pp. 59–62, incorporated herein by reference.

A recommendation engine 600 has two roles and two interfaces that match these roles: recording customer preferences 601 and making recommendations 610. This invention does not change the preference recording role, which is accomplished through the recommendation engine's rating interface 604. The recommendation process includes compatibility as part of the recommendation criteria.

A system using the modified recommendation engine has the following components: a ratings input interface 604; a recommendation request interface 605, which supports requests for recommendations for specific items or requests for a set of recommended items; an optional item match set input 602, which is used with a request for recommendations to specify a set of items with which the recommendation set should be compatible; an item compatibility rule set input 606; a recommendation process 608, storage 609 for customer ratings of items; and a recommendation output set 610, which is the output resulting from a recommendation request 603.

The match set input 602 plays a role similar to that of the shopping set 202 and historical set 203 described in the first embodiment. The system 600 is referred to as a compatibility–aware recommendation engine.

Implementations of a compatibility–aware recommendation engine 600 may include any of the methods described above with regard to the first embodiment of FIGs. 2–5, with the shopping set 202 replaced by the match set 602 and the historical set not provided. An advantage of the present embodiment is that closer connections may be made between the recommendation process and the compatibility information.

The compatibility process may be integrated into the recommendation in one of at least three ways:

1. Use of a match set and item compatibility rules to drive the recommendation process.
2. Use of the recommendation engine to enhance item compatibility algorithms.
3. Integration of item compatibility and recommendation processes in the recommendation engine.

One particular method that uses the match set and item compatibility rules to drive the recommendation process is shown in FIGs. 7A and 7B. The method 750 illustrated in FIG. 7A is a generalized method, whereas the method 700 in FIG. 7B is one particular implementation of the generalized method 750. The following description relates the specific steps shown in FIG. 7B to the general steps of FIG. 7A. In this method 750, a compatibility–aware recommendation set 610 is produced using the following steps:

1. A set of items compatible with those in the match set is generated, at step 752. The set of compatible items includes those that complement items in the match set but do not substitute for items in the match set. This is a generalized description of steps 703–709.
2. Recommendation scores are obtained for each item in the set, step 754. This is a generalized description of steps 711–713.
3. A subset of those items is selected such that no item and its substitute are in the set and such that more highly recommended items outweigh items with lower recommended scores, at step 756. This is a generalized description of steps 714 – 723.

Turning now to the specific embodiment illustrated in FIG. 7B, it is important to note that all sets are assumed to be ordered sets without replication. The UNION operator is assumed to preserve order. Various working sets, **RecSet**, and **OutSet** are initialized at step 701, and a determination is made, at step 702, whether the match set 602, **MatchSet**, exists. If there is no match set 602, then the method passes directly to step 710. If there is a match set 602, then the set **MatchSet** is tested to determine whether it is empty, at step 703. If it is empty, then the method passes to step 709. If **MatchSet** is not empty, then the first item in **MatchSet** is selected out as

Mitem and a temporary copy of rules **Temp** is initialized, in step 704. After **Temp** is tested for being empty, at step 705, the first rule in **Temp** is extracted as **Rule**, at step 706. If **Rule** is a complement rule and **Mitem** is on the left hand side of the **Rule**, as determined at step 707, then the item on the right hand side of **Rule** is added to **RecSet**, at step 708. Once all the rules have been tested, as determined at step 705, the method returns to step 703 for the next item in **MatchSet**.

Once it has been determined, at step 703, that no items remain in **MatchSet**, a determination is made whether **RecSet** is empty. If it is empty, then the method proceeds to step 710, where **OutSet** is set equal to **RecommendedNItems**, at step 710 and output at step 724.

If **RecSet** is not empty, then a loop, steps 711–713, is set up to produce a recommendation score for each item in **RecSet**. Once all items have recommendation scores, the items in **RecSet** are sorted in descending order according to recommendation score, at step 714.

The first item in **RecSet** is extracted as **Ritem**, a temporary copy of the rules is initialized as **TRule**, and a temporary copy of **OutSet** is set up as **TRec**, at step 716. A determination is made, at step 717, as to whether **TRule** is empty. If so, then **Ritem** is added to **OutSet** via the UNION operator, at step 723. If not, then the first rule in **TRule** is extracted as **Rule**, at step 718. A determination is made, at step 719, whether **Rule** is a substitute rule and **Ritem** is on the right hand side. If not, then the method returns to step 717. If so, then a determination is made whether **TRec** is empty, at step 720. If so, then the method returns to step 717. If not, the first item in **TRec** is extracted as **Rec**, at step 721, and a determination made, at step 722, whether **Rec** is on the left hand side of **Rule**. If **Rec** is not on the left hand side of **Rule**, then the method returns to step 720. On the other hand, if **Rec** is on the left hand side of **Rule**, then the method proceeds to step 715.

A method 800 that bears some similarity to the method 500 illustrated in FIG. 5 takes advantage of a recommendation engine's prediction abilities, and is in FIG. 8. In this method 800, the "Apply a Rule" step 810 generates a recommendation value within the recommendation engine, rather than starting new items in the list with a neutral default recommendation value. This illustrates an advantage of this second

embodiment, *viz.* the ability to transition seamlessly between the recommendation process and the item compatibility process.

FIG. 9 shows a data structure 900 and methods 901, 902 for inferring item compatibility rules in a recommendation engine. The data structure 900 is the upper right triangle of a two-dimensional array that maps items to items. In each cell of the array is stored the number of times that each pair of items was purchased together, either as part of a single transaction or as part of a longer-term customer relationship. This data is augmented by a simple array 903 that stores the total number of transactions, N_{aa}, N_{ab},....N_{zz} (or relationships) in which each item, aa, ab,.....zz, was purchased.

The substitute rule inference method 901 finds pairs of items such that one item is generally not purchased when the other is purchased. It has two operating thresholds, one for the maximum co-purchase percentage and another for the number of purchases needed to judge accurately. The maximum percentage (**SubThreshold**) identifies when items are likely not to be substitutes. The number of purchases (**SubSignifThreshold**) is an estimate of confidence. For example, we may decide that groceries require a maximum co-purchase percentage of 1% and a minimum of 10,000 purchases to judge accurately. If, for example, we are testing skim milk against whole milk, and skim milk was purchased 20,000 times, whole milk was purchased 15,000 times, and they were purchased together 175 times, we would conclude as follows:

1. When people buy skim milk, they generally don't buy whole milk (175/20,000 < 0.01) and with 20,000 examples of buying skim milk, we confidently infer the rule "skim milk -> NOT whole milk".
2. When people buy whole milk, they buy skim milk often enough to prevent a rule (175/15,000 > 0.01).

The complement rule inference method 902 finds pairs of items such that one item is generally purchased when the other is purchased. It has two operating thresholds, one for the minimum co-purchase percentage and another for the number of purchases needed to judge accurately. The minimum percentage (**CompThreshold**) identifies when items are likely not to be complements. The number of purchases

(CompSignifThreshold) is an estimate of confidence. For example, we may decide that groceries require a minimum co-purchase percentage of 50% and a minimum of 10,000 purchases to judge accurately. If, for example, we are looking at the correlation between purchases of breakfast cereal and milk, and milk was purchased 20,000 times, while cereal was purchased 15,000 times, and they were purchased together 8,000 times, we would conclude as follows:

1. When people buy cereal , they generally buy milk (8,000/15,000 ≥ 0.5) and with 15,000 examples of buying milk, we confidently infer the rule "cereal -> milk".
2. When people buy milk, they also buy cereal sufficiently infrequently that we are prevented from inferring a rule (8,000/20,000 > 0.5).

Alternative implementations of these algorithms can use basic statistical methods to vary the number of items needed for confidence. In this case, the parameter used is the desired probability of correctness. Also, while the example uses language of purchases, the algorithms and data structures can learn from non-purchase behavioral data including but not limited to reading items, evaluating items, and rating items.

The present invention is not restricted to operating on a single processor system. In the embodiments of the invention described herein, each process may reside wholly within a single processor or be distributed among multiple processors. For example, the recommendation engine 608 may itself be resident on a single processor or distributed among several processors either for economic advantage, to achieve redundancy and higher availability, to achieve better performance, for geographic diversity, or for other reasons. Furthermore, each separate process may operate on the same processor(s), different processor(s), or an overlapping but different set of processors from each other process. For example, with regard to the embodiment illustrated in FIG. 2, an item compatibility rule input may be on a different processor from a compatibility modifier, but on the same processor as the historical set input, while the shopping set input may be split between the processor of the compatibility modifier and a third processor. In the case where more than one processor is used in the system, the processors are constructively, or operatively, coupled. Methods of such

constructive coupling include, but are not limited to, connection through the internal bus of a multiprocessor computer, connection through shared computer memory or storage devices, connection on a local area network, connection on a wide area network, connection through as-needed communications such as modems and telephone lines, or connection through periodic data interchange such as transferring data on disks or tapes.

The present invention is useful in many different applications. For convenience, we refer to the output obtained from the two embodiments described above as "compatibility modified recommendations," to the first embodiment 200 as a "compatibility modifier for recommendations," and to the second embodiment 600 as a "compatibility aware recommendation engine." The use of the term "compatibility modified recommendations" is not intended to imply interest only in the output recommendation set, but rather to refer to the entire method that generates such output recommendation sets.

One way of developing an application that uses compatibility modified recommendations, a system designer or administrator identifies and connects the source of rules, determines whether the system will use shopping, historical, or match sets, and configures the recommendation system as it would be used without compatibility modification. The designer or administrator then substitutes the compatibility aware recommendation engine for an ordinary recommendation engine or adds a compatibility modifier for recommendations between the recommendation generation and its display to a user. Figure 10 shows one embodiment of how these modifications may be made to an existing system. System 1080 is a non-compatibility-aware call center system having a call center agent console 1051 connected, via a network 1052, to a call center application server 1053. The application server 1053 is further connected to one or more memory units containing customer information 1054 and product information 1055. The application server 1053 is also connected to receive a list of items 1056 recommended for marketing to customers. The list may be generated using any of the techniques listed above.

System 1090 is a call center system that includes compatibility filtering. Instead of receiving the list of recommendations 1056 directly, the application server

1053 receives compatibility filtered and weighted recommendations 1064 from a compatibility filter 1061. The compatibility filter receives 1061 the list of recommendations 1056 and, using compatibility rules 1063, current purchase information 1061 from the application server 1053, and customer history information 1062 from the customer information 1054, filters and weights the recommendations to produce the set of compatibility filtered and weighted recommendations 1064.

While each application has different requirements, we present two sample applications and also identify a larger number of applications of the invention. This set of examples is not intended to limit the applicability of the invention.

First, the present invention may be extremely useful in electronic commerce applications on the Internet. FIG. 11 shows an illustration of a compatibility-aware recommendation engine 1110 in an electronic commerce application. The customer 1102 uses a web browser, for example on a personal computer 1101 to connect through a network 1101 to a web server 1104. The commerce server 1106, connected to the web server 1104, processes all financial transactions for the customer 1102 and contains a database of products for sale. The web server 1104 presents this set of products for sale to the customer 1102. A purchase database 1108, coupled to the commerce server 1106, contains records of previous purchases made by the customer 1102 and other customers.

A compatibility-aware recommendation engine 1101 is coupled to the web server 1104, the commerce server 1106, and the purchase database 1108. The recommendation engine 1101 may purchase data, e.g. implicit ratings, from the purchase database 1108 and may also receive explicit ratings from the customer 1110 via the web server 1104. The recommendation engine 1106 also receives match set information from the commerce server 1106, i.e. the current state of the market basket and any other relevant customer historical items. The recommendation engine 1110 receives requests from the web server 1104 and/or the commerce server 1106 for recommendations. These requests may focus on specific products or may be more general. The recommendation engine 1110 generates recommendations, using

compatibility information, to ensure that the recommendations supplied are coherent and useful as a set and consistent with the user's current purchases.

There are many content domains within electronic commerce where a compatibility aware recommendation engine, or compatibility modified recommendations in general, would be useful. A specific example is an on-line bookstore. In this application, the commerce server contains a database of all books available for sale, indexed at least by ISBN, title, author, and subject. The commerce server also maintains a "shopping basket" of books that the user is currently planning to buy and a set of "recently purchased" books that the user has purchased within certain period of time, for example the past six months. The web server presents the user with various options for browsing books including viewing by topic, searching for books based on author, title, or ISBN, browsing best sellers, or simply asking for recommendations. Whenever a specific book or small set of books is sought, the web server can request recommendation scores from the compatibility aware recommendation engine for recommended books and present the recommendations to the user. These recommendation scores may be compatibility modified to help the user and the system identify books that match well with other purchases. When the user requests recommendations or a large set of books such as "fiction," the web server may ask the compatibility-aware recommendation engine for recommendations overall or within a category. The recommendation engine can then evaluate the books both on their own merits for the user, and as part of a set of books, and recommend a coherent set of books that is compatible with the other books selected by the user.

Another use of the recommendation engine is for the web server to request recommendations for books to advertise as part of the check-out process. The invention is advantageous because it prevents the system from recommending substitutes for current purchases and suggests books that complement the purchases.

A second illustrative application for such a system is to provide support for human sales staff in suggestive selling. In this application, depicted in FIG. 12, a sales agent 1214 uses an interface 1206 to a call center console 1204. The interface 1206 is used by the agent 1214 to enter product requests from the customer 1202. The

call-center console 1204 is coupled to a commerce server 1208 that contains information regarding products on offer to the customer 1202, typically including pricing and availability. A purchase database 1210 is connected to the commerce server 1208 to record transactions with the customer 1202 and other customers. A recommendation engine 1212, either a recommendation engine having compatibility awareness integrated with the recommender, or having a compatibility modifier to modify recommendations, is coupled to the commerce server 1208 and the purchase database 1210. Compatibility-aware or compatibility-modified recommendations are generated by the recommender system 1212, and are updated to reflect each customer product request. the recommendations are displayed to the sales agent 1214 to help suggest other products for cross-selling. Depending on the business objectives of the marketer, the recommendations may be limited to a set of products that are featured, may span the inventory of the business, or may be restricted to products with specific attributes, for example products that are in-stock or have a high mark-up.

Another example is illustrated in FIG. 13 that shows a cash-register check-out system. A cash register 1302 is connected to via a recommender system 1306 to a purchase database 1308. The recommender system 1306 suggests additional purchases to the customer while purchases are entered into the cash register 1302. An additional feature of the cash register check-out system is the ability to print a coupon compatible with current purchases using a coupon printer 1304; this coupon may induce the customer to return to make more purchases sooner than the customer would have returned without the coupon.

Many other content areas can similarly be enhanced by compatibility modified recommendations. Examples include music in various forms (e.g., sheet music, music recordings, music video recordings, and on-demand music systems including jukeboxes and cable-TV music request services), advertisements, marketing literature and product offers, consumable goods including groceries and office supplies, dining and entertainment services (e.g., arranging a set of dinner and/or theater reservations that complement each other), financial service products (e.g., recommending financial service products compatible with the customer's goals and

portfolio), real estate and home furnishings, automobile-related goods and services, travel-related goods and services, media of various forms (e.g., audio, video, images), computer products and services, art works, publications and documents. In each of these areas, and many others; the benefit of compatibility modified recommendations is the added value to the customer and to the marketer when product recommendations are consistent with prior purchases, concurrent purchases, and each other.

Another embodiment of the invention, illustrated in FIG. 14A, includes a processor system 1400 configured to receive applicable data including i) item recommendations 1406, ii)a shopping set 1408 and iii) a history set 1410. The processing system is configured to produce a set of item compatibility rules 1404. The processing system 1400 is also configured to produce a compatibility-modified recommendation set 1412 using at least the item recommendations 1405 and the item compatibility rules 1404. The processor system 1400 may be a single processor for producing the compatibility rules and performing the compatibility modification.

The processing system may also include a number of different processors. Another embodiment of a the processing system is illustrated in FIG. 14B, where the processing system includes a first processor 1420 (shown in dashed lines) configured to receive the recommendation set 1406, the shopping set 1408 and the history set 1410, and to modify the recommendations to become compatibility aware. A second processor 1422 (also in dashed lines) is operatively coupled to the first processor 1420 and is configured to produce the item compatibility rules 1404 and to direct the item compatibility rules to the first processor 1420. The second processor 1422 may be remote from the first processor 1420 and coupled to the first processor 1420 through a network , the Internet, or some other communications channel.

It will be appreciated that the compatibility-aware recommender system illustrated in FIG. 6 may be situated on processing system having a single processor, or a number of processors. For example, the recommendation process 608 may be on a first set of processors, including a single processor, while all other items, including the item compatibility rules 606, match set 602, rating storage 609, and other items, are all located on a second set of processors, including a single processor. In another example,

the recommendation process 608 may be on a first set of processors, while all the other items, including the item compatibility rules 606, the match set 602, and the rating storage 609, are each located on, or associated with, their own separate, individual set of processors. It should be appreciated that where the term "set of processors" is used, the set may include only a single processor.

As noted above, the present invention is applicable to recommender systems. It is believed to be particularly useful in permitting recommender systems to produce recommendations that are not only accurate but also of high value to the user, i.e. recommendations are compatible with other purchases or recommendations. Accordingly, the present invention should not be considered limited to the particular examples described above, but rather should be understood to cover all aspects of the invention as fairly set out in the attached claims. Various modifications, equivalent processes, as well as numerous structures to which the present invention may be applicable will be readily apparent to those of skill in the art to which the present invention is directed upon review of the present specification. The claims are intended to cover such modifications and devices.

WE CLAIM:

1. An electronic processing system for generating a compatibility-modified recommendation output set to a user based, at least in part, on a set of item compatibility rules, the system comprising a processing system of one or more sets of processors configured to:

a. receive applicable data including

i. item recommendation data, and

ii. item compatibility rules, and

b. modify the item recommendation data using the item compatibility rules to produce a compatibility-modified recommendation output set.

2. A system as recited in claim 1, wherein the processing system is further configured to

a. receive shopping data,

b. apply the item compatibility rules to the shopping data, and

c. modify the item recommendation data using the item compatibility rules applied to the shopping data to produce the compatibility-modified recommendation output set.

3. A system as recited in claim 2, wherein shopping data received by the processing system is derived from one of: a customer action of selecting an item for purchase; contents of an electronic "market basket"; active shopping carts; cash register transactions; a customer's shopping list; and observation of customer behavior.

4. A system as recited in claim 1, wherein the processing system is further configured to

a. receive historical data,

b. apply the item compatibility rules to the historical data, and

c. modify the item recommendation data using the item compatibility rules applied to the historical data to produce the compatibility-modified recommendation output set.

5. A system as recited in claim 4, wherein the historical data is received as one of: prior purchases; recent purchases; expressed or inferred user preferences; data gathered from credit card records; and data gathered from marketing surveys.

6. A system as recited in claim 1, wherein the processing system is configured to receive the item recommendation data from at least one of: an outside source; a genetic algorithm; a collaborative filter; a neural network; a statistical model; a vendor specification; a human expert; and a computer–based expert.

7. A system as recited in claim 1, wherein the processing system receives the item recommendation organized as one of: an unordered set; an ordered set without recommendation values; and an ordered set with recommendation values.

8. A system as recited in claim 1, wherein the processing system is further configured to derive the item compatibility rules from one or more of:

a. a marketer specification,
b. automatic generation using machine learning,
c. automatic generation using a genetic algorithm,
d. automatic generation using a neural network,
e. automatic generation using a rule inference system,
f. data mining,
g. an analysis of historical purchase and preference data, and
h. a customer specification.

9. A system as recited in claim 1, wherein the processing system is further configured to receive the item compatibility rules represented as at least one of:

a. unidirectional rules,

b. bidirectional rules,

c. generalized rules including multi-way rules,

d. rules among items,

e. rules among sets,

f. rules among collections,

g. rules with weight factors,

h. rules with priorities, and

i. unweighted and unprioritized rules.

10. A system as recited in claim 1, wherein the processing system is further configured to:

a. determine whether a shopping set exists,

b. receive the shopping set when the shopping set is determined to exist,

c. determine whether a history set exists,

d. receive the history set when the history set is determined to exist,

e. receive the item compatibility rules, and

g. modify the recommendation set based on the received item compatibility rules and the shopping set and the history set when the shopping set and the history set are determined to exist.

11. A system as recited in claim 1, wherein the processing system is further configured to:

a. add items in the item recommendation data to the compatibility-modified recommendation output set,

b. add, to the compatibility-modified recommendation output set, items that complement items in the item recommendation data, and

c. remove, from the compatibility-modified recommendation output set, substitute items that are substitutes for items placed in the modified

recommendation set before the substitute items were placed in the modified recommendation set.

12. A system as recited in claim 11, wherein the processing system is further configured to:

a. determine whether a shopping set exists,

b. initialize the compatibility-modified recommendation set with items that complement items in the shopping list when the shopping set is determined to exist, and

c. remove, from the compatibility-modified recommendation output set, items that are substitutes for items in the shopping set.

13. A system as recited in claim 11, wherein the processing system is further configured to:

a. determine whether an historical set exists, and

b. add, to the compatibility-modified recommendation output set, items that complement items in the historical set when the historical set is determined to exist.

14. A system as recited in claim 1, wherein the item recommendation data includes recommendation values, and the processing system is further configured to:

a. initialize the compatibility-modified recommendation set with the item recommendation data,

b. determine whether a shopping set exists and whether an historical set exists,

c. for each particular item in the shopping set, when the shopping set is determined to exist, apply rules having the particular item in the shopping set on the rules' inputs, and add items to the compatibility-modified recommendation set and subtract items from the compatibility-modified recommendation set when the applied rules so determine,

d. for each particular item in the historical set, when the historical set is determined to exist, apply rules having the particular item in the historical set on the rules' inputs, when the particular item in the historical set is not also in the shopping set, and add items to the compatibility-modified recommendation set and subtract items from the compatibility-modified recommendation set when the applied rules so determine,

e. sort items in the compatibility-modified recommendation output set according to recommendation value,

f. for each examined item whose recommendation score exceeds a recommendation value, select a first sub-set of rules whose inputs include the examined item,

g. select a second sub-set of rules from the first sub-set of rules where rules in the second sub-set have outputs with items not appearing above the examined item in the compatibility-modified output set,

h. apply the rules in the second sub-set,

i. sort items in the compatibility-modified recommendation output set below the examined item according to recommendation score, and

j. sort all items in the compatibility-modified recommendation output set according to recommendation score.

15. A system as recited in claim 14, wherein applying a rule to a selected item includes

examining a rule having the selected item on an input,

adding an item on an output of the rule to the compatibility-modified recommendation output set where the item on the output is not already in the compatibility-modified recommendation output set, and applying a neutral recommendation score to the added item, and

adding a recommendation score modifier to a recommendation score for all items on the output of the rule.

16. A system as claimed in claim 1, further configured to:

a. determine that a first and a second item have each been selected a number of times that is greater than a selection threshold,

b. determine the number of times the first item has been selected by the user when the user has selected the second item,

c. compare, for one of the first and second items, the number of times that the one of the first and second items has been selected at the same time as the other of the first and second items, with the total number of times the one of the first and second items has been selected to produce a coincidence indicator, and

d. infer a substitute rule between the two items when the coincidence indicator indicates coincident selection of the first and second items at a level below a substitute threshold level.

17. A system as claimed in claim 1, further configured to:

a. determine that a first and a second item have each been selected a number of times that is greater than a selection threshold,

b. determine the number of times the first item has been selected by the user when the user has selected the second item,

c. compare, for one of the first and second items, the number of times that the one of the first and second items has been selected at the same time as the other of the first and second items, with the total number of times the one of the first and second items has been selected to produce a coincidence indicator, and

d. infer a complementary rule between the two items when the coincidence indicator indicates coincident selection of the first and second items at a level above a coincidence threshold level.

18. A system as recited in claim 1, wherein the processing system comprises a first set of processors connected to a computer network, the set of first processors being configured to

a. receive the applicable data including

i. the item recommendation data, and

ii. the item compatibility rules, and

b. modify the item recommendation data using the item compatibility rules to produce the compatibility-modified recommendation output set.

19. A system as recited in claim 18, wherein the processing system further includes a second set of processors connected to the computer network, the second set of processors being configured to derive the item compatibility rules.

20. A system as recited in claim 18, wherein the first set of processors is configured to derive the item compatibility rules.

21. A method of producing a compatibility filtered and weighted recommendation to a user, the method using a computer having a processing system having one or more sets of processors; and an input/output interface, the method comprising:

a. receiving, by the processing system, applicable data including

i. item recommendation data, and

ii. item compatibility rules, and

b. modifying the item recommendation data, with the processing system, using the item compatibility rules to produce a compatibility-modified recommendation output set.

22. A method as recited in claim 21, further comprising

a. receiving shopping data, using the processor system, from the input/output interface,

b. applying the item compatibility rules to the shopping data using the processor system, and

c. modifying the item recommendation data, with the processing system, using the item compatibility rules applied to the shopping data to produce the compatibility-modified recommendation output set.

23. A method as recited in claim 22, further comprising receiving shopping data, using the processor system, derived from one of: a customer action of selecting an item for purchase; contents of an electronic "market basket"; active shopping carts; cash register transactions; a customer's shopping list; and observation of customer behavior.

24. A method as recited in claim 21, further comprising

a. receiving historical data using the processor system,

b. applying the item compatibility rules to the historical data using the processor system, and

c. modifying the item recommendation data, with the processor system, using the item compatibility rules applied to the historical data to produce the compatibility-modified recommendation output set.

25. A method as recited in claim 24, further comprising receiving the historical data, using the processor system, as one of: prior purchases; recent purchases; expressed or inferred user preferences; data gathered from credit card records; and data gathered from marketing surveys.

26. A method as recited in claim 21, further comprising receiving item recommendation data, using the processor system, from at least one of an outside source, a genetic algorithm, a collaborative filter, a neural network, a statistical model, a vendor specification, a human expert, and a computer–based expert.

27. A method as recited in claim 21, further comprising receiving the item recommendation data, using the processor system, organized as one of an unordered set,

an ordered set without recommendation values, and an ordered set with recommendation values.

28. A method as recited in claim 21, further comprising deriving the item compatibility rules, using the processor system, from one or more of:

a. a marketer specification,
b. automatic generation using machine learning,
c. automatic generation using a genetic algorithm,
d. automatic generation using a neural network,
e. automatic generation using a rule inference system,
f. data mining,
g. an analysis of historical purchase and preference data, and
h. a customer specification.

29. A method as recited in claim 21, further comprising receiving, using the processor system, the item compatibility rules represented as at least one of:

a. unidirectional rules,
b. bidirectional rules,
c. generalized rules including multi-way rules,
d. rules among items,
e. rules among sets,
f. rules among collections,
g. rules with weight factors,
h. rules with priorities, and
i. unweighted and unprioritized rules.

30. A method as recited in claim 21, further comprising

a. determining, using the processor system, whether a shopping set exists,

b. receiving, using the processor system, the shopping set when the shopping set is determined to exist,

c. determining, using the processor system, whether a history set exists,

d. receiving, using the processor system, the history set when the history set is determined to exist,

e. receiving, using the processor system, the item compatibility rules, and

g. modifying, using the processor system, the recommendation set based on the received item compatibility rules and the shopping set and the history set when the shopping set and the history set are determined to exist.

31. A method as recited in claim 21, further comprising:

a. adding items in the item recommendation data, using the processor system, to the compatibility-modified recommendation output set,

b. adding, to the compatibility-modified recommendation output set, items that complement items in the item recommendation data, using the processor system, and

c. removing, from the compatibility-modified recommendation output set, substitute items that are substitutes for items placed in the modified recommendation set before the substitute items were placed in the modified recommendation set, using the processor system.

32. A method as recited in claim 31, further comprising

a. determining, using the processor system, whether a shopping set exists,

b. initializing the compatibility-modified recommendation set with items that complement items in the shopping list when the shopping set is determined to exist, using the processor system, and

c. removing, from the compatibility-modified recommendation output set, items that are substitutes for items in the shopping set, using the processor system.

33. A method as recited in claim 31, further comprising

a. determining, using the processor system, whether an historical set exists, and

b. adding, to the compatibility-modified recommendation output set, items that complement items in the historical set when the historical set is determined to exist, using the processor system.

34. A method as recited in claim 21, wherein the item recommendation data includes recommendation values, and further comprising

a. initializing, using the processor system, the compatibility-modified recommendation set with the item recommendation data,

b. determining whether a shopping set exists and whether an historical set exists, using the processor system,

c. for each particular item in the shopping set, when the shopping set is determined to exist, applying rules having the particular item in the shopping set on the rules' inputs, and adding items to the compatibility-modified recommendation set and subtracting items from the compatibility-modified recommendation set when the applied rules so determine, using the processor system,

d. for each particular item in the historical set, when the historical set is determined to exist, applying rules having the particular item in the historical set on the rules' inputs, when the particular item in the historical set is not also in the shopping set, and adding items to the compatibility-modified recommendation set and subtracting items from the compatibility-modified recommendation set when the applied rules so determine, using the processor system,

e. sorting items in the compatibility-modified recommendation output set according to recommendation value, using the processor system,

f. for each examined item whose recommendation score exceeds a recommendation value, selecting a first sub-set of rules whose inputs include the examined item, using the processor system,

g. selecting, using the processor system, a second sub-set of rules from the first sub-set of rules where rules in the second sub-set have outputs with items not appearing above the examined item in the compatibility-modified output set,

h. applying the rules in the second sub-set, using the processor system,

i. sorting items in the compatibility-modified recommendation output set below the examined item according to recommendation score, using the processor system, and

j. sorting all items in the compatibility-modified recommendation output set according to recommendation score, using the processor system.

35. A method as recited in claim 34, wherein applying a rule to a selected item includes

examining a rule having the selected item on an input, using the processor system,

adding, using the processor system, an item on an output of the rule to the compatibility-modified recommendation output set where the item on the output is not already in the compatibility-modified recommendation output set, and

applying, using the processor system, a neutral recommendation score to the added item, and

adding, using the processor system, a recommendation score modifier to a recommendation score for all items on the output of the rule.

36. A method as claimed in claim 21, further comprising

a. determining, using the processor system, that a first and a second item have each been selected a number of times that is greater than a selection threshold,

b. determining, using the processor system, the number of times the first item has been selected by the user when the user has selected the second item,

c. comparing, for one of the first and second items, the number of times that the one of the first and second items has been selected at the same time as the other of the first and second items, with the total number of times the one of the first and second items has been selected to produce a coincidence indicator, using the processor system, and

d. inferring, using the processor system, a substitute rule between the two items when the coincidence indicator indicates coincident selection of the first and second items at a level below a substitute threshold level.

37. A method as claimed in claim 21, further comprising

a. determining, using the processor system, that a first and a second item have each been selected a number of times that is greater than a selection threshold,

b. determining, using the processor system, the number of times the first item has been selected by the user when the user has selected the second item,

c. comparing, for one of the first and second items, the number of times that the one of the first and second items has been selected at the same time as the other of the first and second items, with the total number of times the one of the first and second items has been selected to produce a coincidence indicator, using the processor system, and

d. inferring, using the processor system, a complementary rule between the two items when the coincidence indicator indicates coincident

selection of the first and second items at a level above a coincidence threshold level.

38. A method as recited in claim 21, wherein the processing system comprises a first set of processors connected to a computer network, and further comprising

a. receiving, using the first set of processors, the applicable data including

i. the item recommendation data, and

ii. the item compatibility rules, and

b. modifying, using the first set of processors, the item recommendation data using the item compatibility rules to produce the compatibility-modified recommendation output set.

39. A method as recited in claim 38, wherein the processing system includes a second set of processors connected to the computer network, and further comprising deriving the item compatibility rules, using the second set of processors.

40. A method as recited in claim 38, further comprising deriving the item compatibility rules, using the first set of processors.

41. A computer-readable program storage device, having a set of program instructions physically embodied thereon, executable by a computer, to perform a method of producing a compatibility filtered and weighted recommendation, the method comprising:

a. receiving applicable data including

i. item recommendation data, and

ii. item compatibility rules, and

b. modifying the item recommendation data using the item compatibility rules to produce a compatibility-modified recommendation output set.

42. A storage device as recited in claim 41, the method further comprising

a. receiving shopping data,

b. applying the item compatibility rules to the shopping data, and

c. modifying the item recommendation data, using the item compatibility rules applied to the shopping data, to produce the compatibility-modified recommendation output set.

43. A storage device as recited in claim 41, further comprising

a. receiving historical data,

b. applying the item compatibility rules to the historical data, and

c. modifying the item recommendation data using the item compatibility rules applied to the historical data to produce the compatibility-modified recommendation output set.

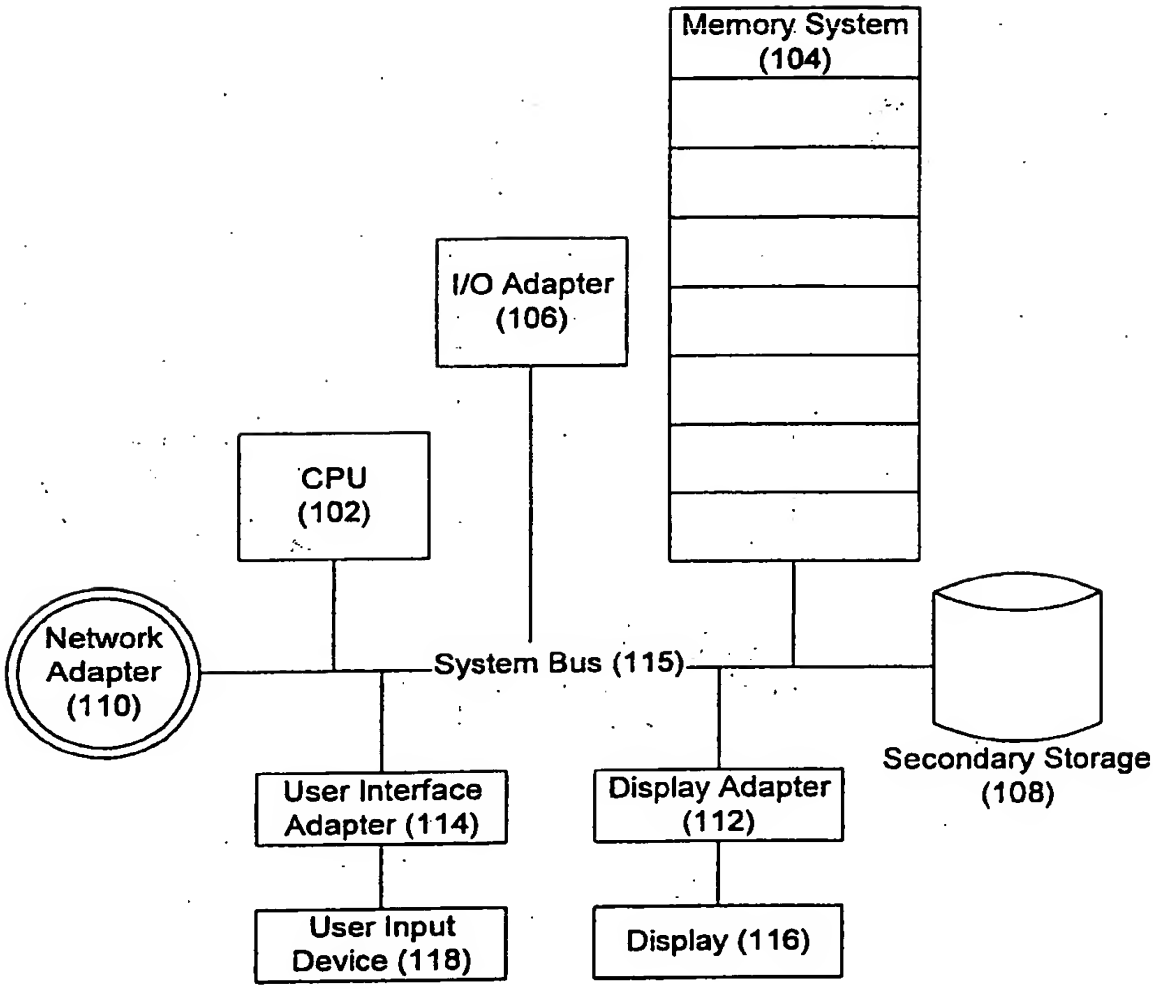

Memory System
(104)
I/O Adapter
(106)
CPU
(102)
Network
Adapter
(110)
System Bus (115)
Secondary Storage
(108)
User Interface
Adapter (114)
Display Adapter
(112)
User Input
Device (118)
Display (116)

FIG. 1

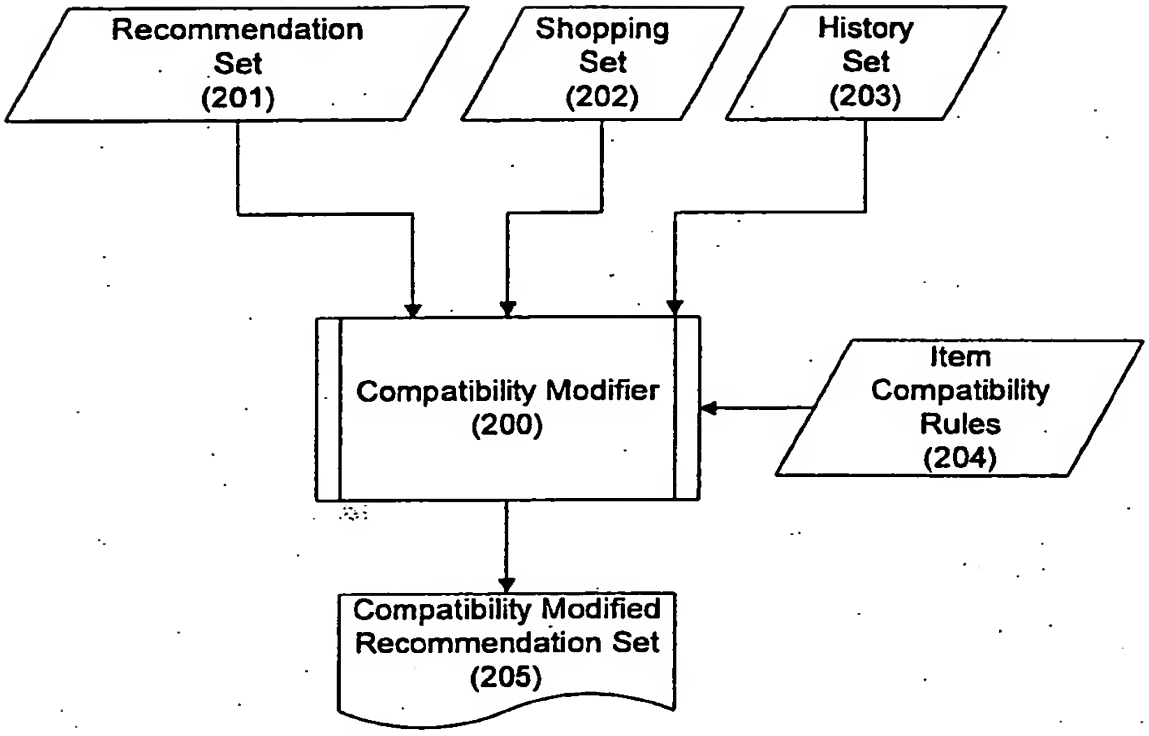

Recommendation Set (201)
Shopping Set (202)
History Set (203)
Compatibility Modifier (200)
Item Compatibility Rules (204)
Compatibility Modified Recommendation Set (205)

FIG. 2

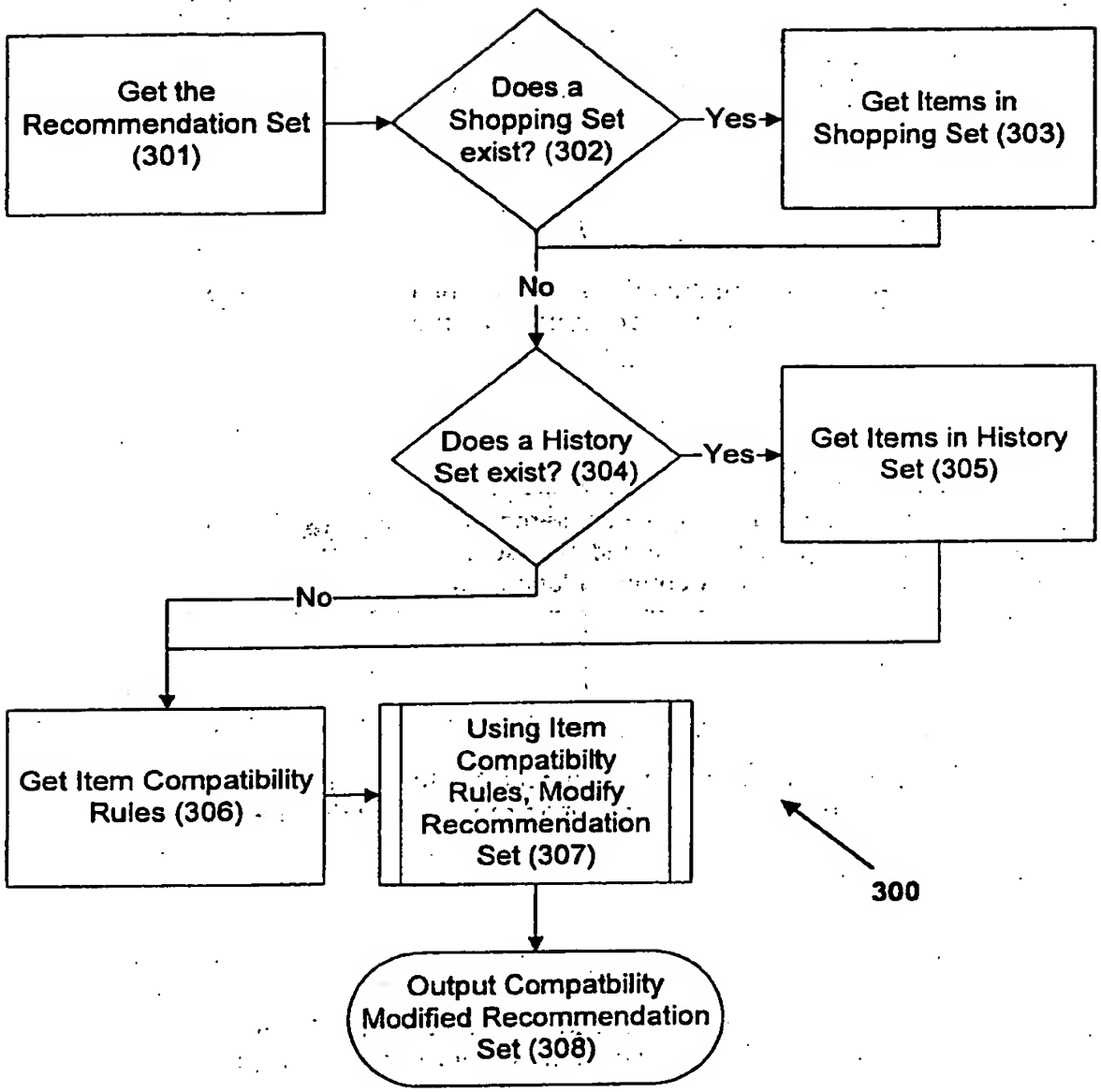

Get the Recommendation Set (301)
Does a Shopping Set exist? (302)
Yes
Get Items in Shopping Set (303)
No
Does a History Set exist? (304)
Yes
Get Items in History Set (305)
No
Get Item Compatibility Rules (306)
Using Item Compatibilty Rules, Modify Recommendation Set (307)
300
Output Compatbility Modified Recommendation Set (308)

FIG. 3

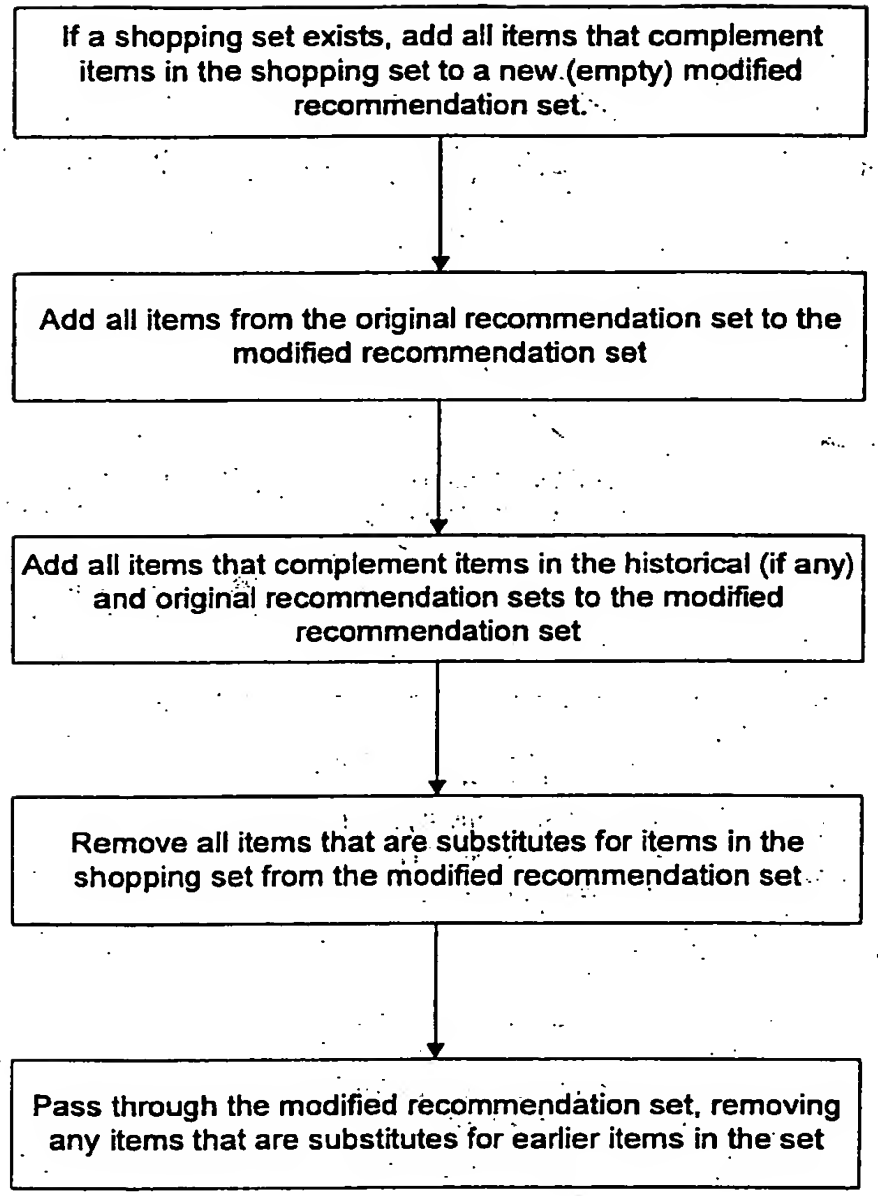

If a shopping set exists, add all items that complement items in the shopping set to a new (empty) modified recommendation set.
Add all items from the original recommendation set to the modified recommendation set
Add all items that complement items in the historical (if any) and original recommendation sets to the modified recommendation set
Remove all items that are substitutes for items in the shopping set from the modified recommendation set
Pass through the modified recommendation set, removing any items that are substitutes for earlier items in the set

FIG. 4A

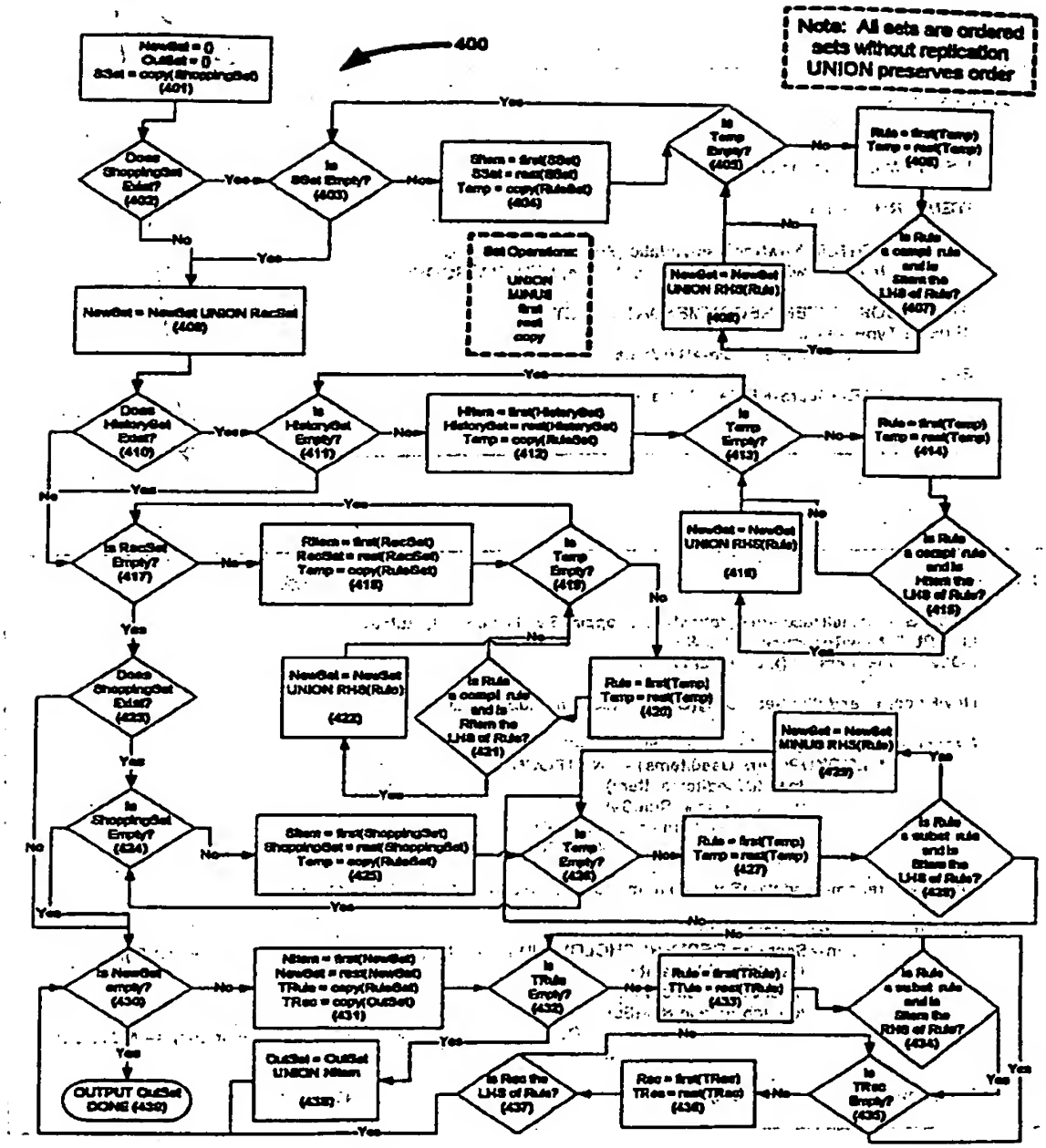

400
Note: All sets are ordered sets without replication UNION preserves order
NewSet = {}
OutSet = {}
SSet = copy(ShoppingSet)
(401)
Does ShoppingSet Exist? (402)
Is SSet Empty? (403)
SItem = first(SSet)
SSet = rest(SSet)
Temp = copy(RuleSet)
(404)
Is Temp Empty? (405)
Rule = first(Temp)
Temp = rest(Temp)
(406)
Is Rule a compl rule and is SItem the LHS of Rule? (407)
NewSet = NewSet UNION RHS(Rule)
(408)
Set Operations:
UNION
MINUS
first
rest
copy
NewSet = NewSet UNION RecSet
(409)
Does HistorySet Exist? (410)
Is HistorySet Empty? (411)
HItem = first(HistorySet)
HistorySet = rest(HistorySet)
Temp = copy(RuleSet)
(412)
Is Temp Empty? (413)
Rule = first(Temp)
Temp = rest(Temp)
(414)
NewSet = NewSet UNION RHS(Rule)
(416)
Is Rule a compl rule and is HItem the LHS of Rule? (415)
Is RecSet Empty? (417)
RItem = first(RecSet)
RecSet = rest(RecSet)
Temp = copy(RuleSet)
(418)
Is Temp Empty? (419)
Does ShoppingSet Exist? (420)
NewSet = NewSet UNION RHS(Rule)
(422)
Is Rule a compl rule and is RItem the LHS of Rule? (421)
Rule = first(Temp)
Temp = rest(Temp)
(423)
NewSet = NewSet MINUS RHS(Rule)
(429)
Is ShoppingSet Empty? (424)
SItem = first(ShoppingSet)
ShoppingSet = rest(ShoppingSet)
Temp = copy(RuleSet)
(425)
Is Temp Empty? (426)
Rule = first(Temp)
Temp = rest(Temp)
(427)
Is Rule a subset rule and is SItem the LHS of Rule? (428)
Is NewSet empty? (430)
NItem = first(NewSet)
NewSet = rest(NewSet)
TRule = copy(RuleSet)
TRec = copy(OutSet)
(431)
Is TRule Empty? (432)
Rule = first(TRule)
TRule = rest(TRule)
(433)
Is Rule a subset rule and is SItem the RHS of Rule? (434)
OUTPUT OutSet
DONE (439)
OutSet = OutSet UNION NItem
(438)
Is Rec the LHS of Rule? (437)
Rec = first(TRec)
TRec = rest(TRec)
(436)
Is TRec Empty? (435)
Yes
No

FIG. 4B

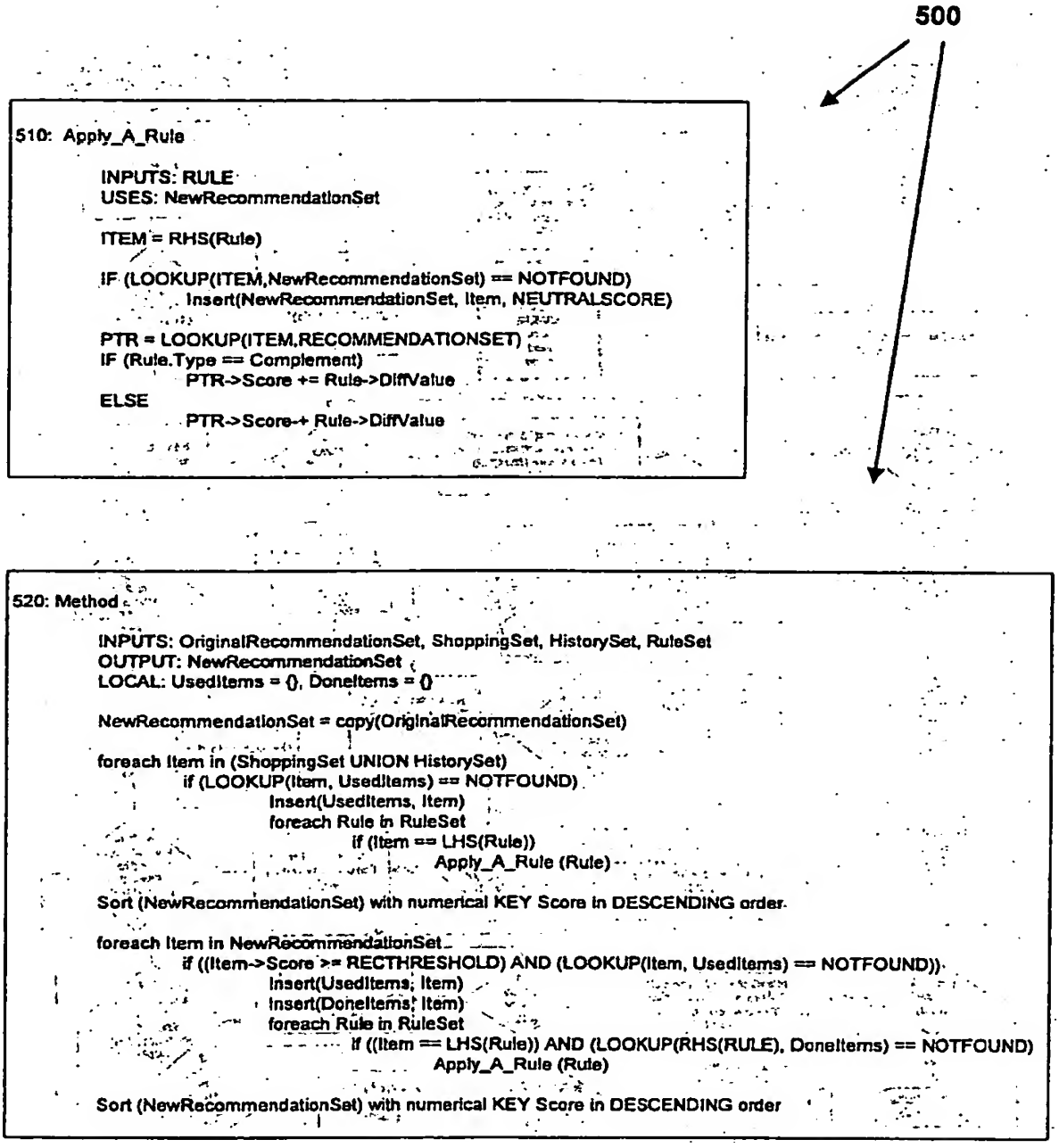

500
510: Apply_A_Rule
INPUTS: RULE
USES: NewRecommendationSet
ITEM = RHS(Rule)
IF (LOOKUP(ITEM,NewRecommendationSet) == NOTFOUND)
Insert(NewRecommendationSet, Item, NEUTRALSCORE)
PTR = LOOKUP(ITEM,RECOMMENDATIONSET)
IF (Rule.Type == Complement)
PTR->Score += Rule->DiffValue
ELSE
PTR->Score-+ Rule->DiffValue
520: Method
INPUTS: OriginalRecommendationSet, ShoppingSet, HistorySet, RuleSet
OUTPUT: NewRecommendationSet
LOCAL: UsedItems = {}, DoneItems = {}
NewRecommendationSet = copy(OriginalRecommendationSet)
foreach Item in (ShoppingSet UNION HistorySet)
if (LOOKUP(Item, UsedItems) == NOTFOUND)
Insert(UsedItems, Item)
foreach Rule in RuleSet
if (Item == LHS(Rule))
Apply_A_Rule (Rule)
Sort (NewRecommendationSet) with numerical KEY Score in DESCENDING order
foreach Item in NewRecommendationSet
if ((Item->Score >= RECTHRESHOLD) AND (LOOKUP(Item, UsedItems) == NOTFOUND))
Insert(UsedItems, Item)
Insert(DoneItems, Item)
foreach Rule in RuleSet
if ((Item == LHS(Rule)) AND (LOOKUP(RHS(RULE), DoneItems) == NOTFOUND)
Apply_A_Rule (Rule)
Sort (NewRecommendationSet) with numerical KEY Score in DESCENDING order

FIG. 5

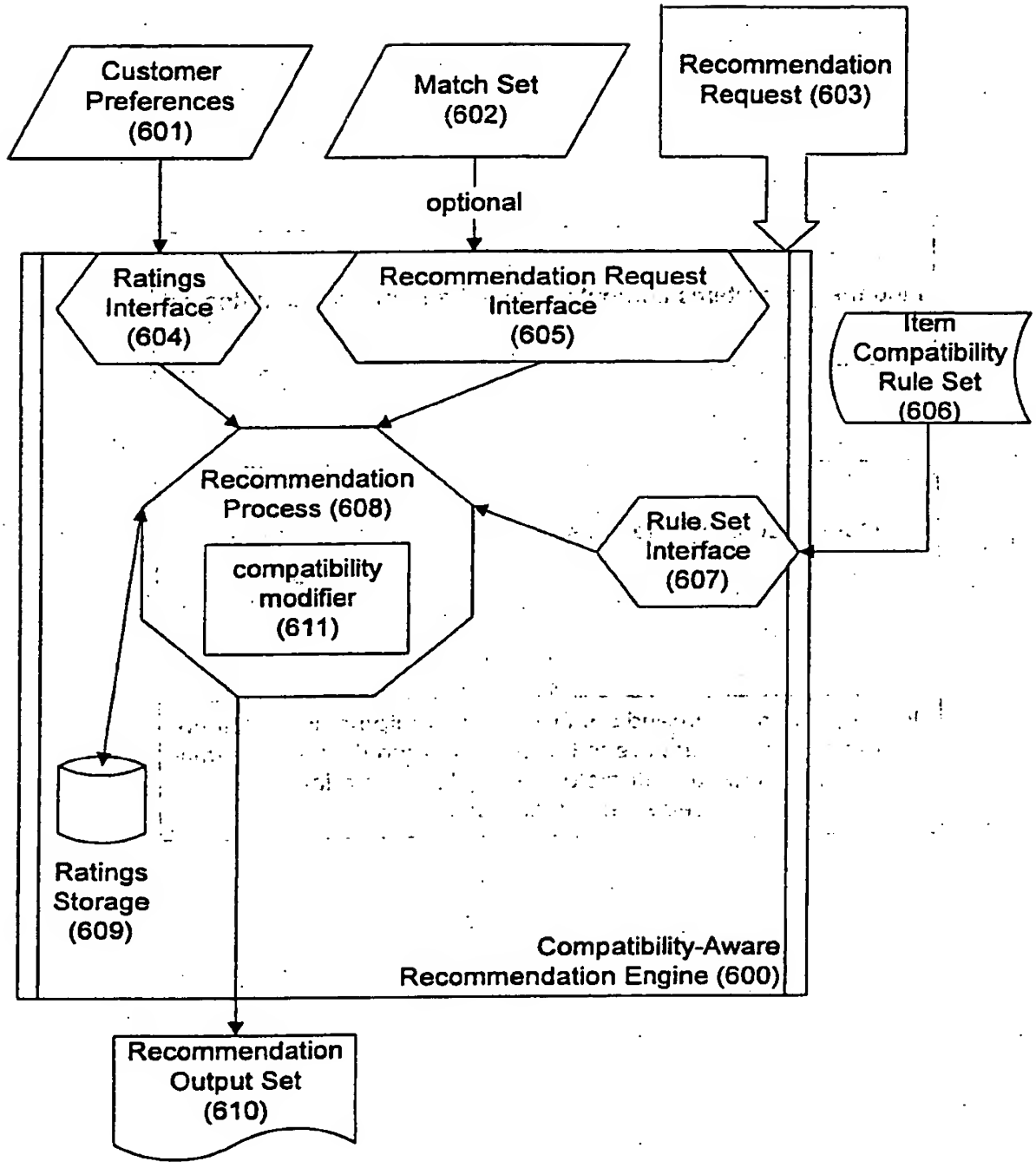

Customer Preferences (601)
Match Set (602)
Recommendation Request (603)
optional
Ratings Interface (604)
Recommendation Request Interface (605)
Item Compatibility Rule Set (606)
Recommendation Process (608)
compatibility modifier (611)
Rule Set Interface (607)
Ratings Storage (609)
Compatibility-Aware Recommendation Engine (600)
Recommendation Output Set (610)

FIG. 6

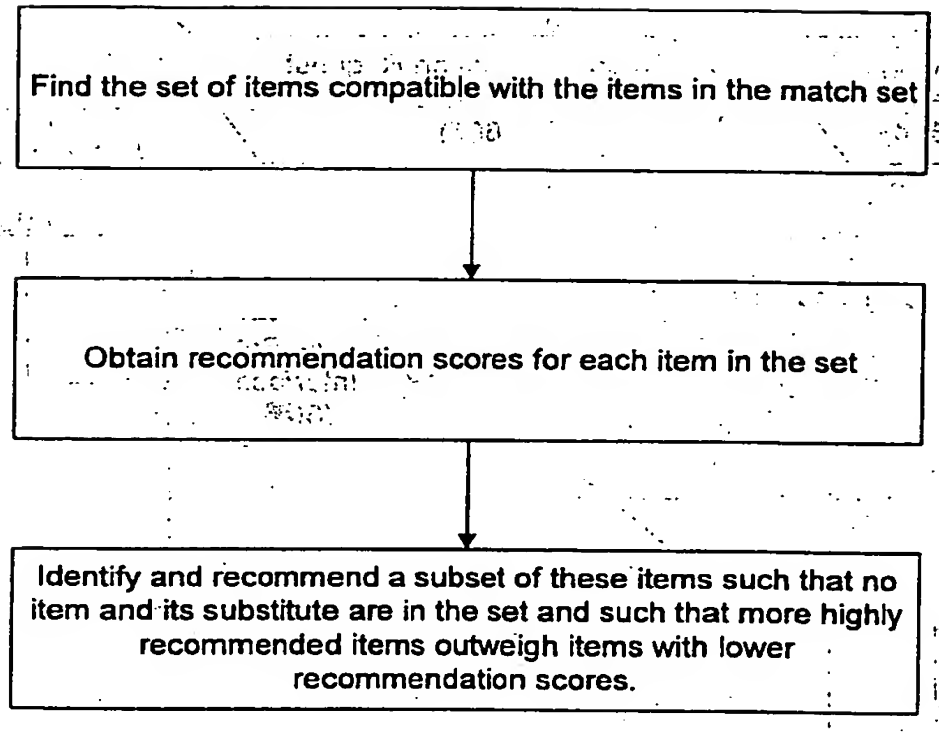

Find the set of items compatible with the items in the match set
Obtain recommendation scores for each item in the set
Identify and recommend a subset of these items such that no item and its substitute are in the set and such that more highly recommended items outweigh items with lower recommendation scores.

FIG. 7A

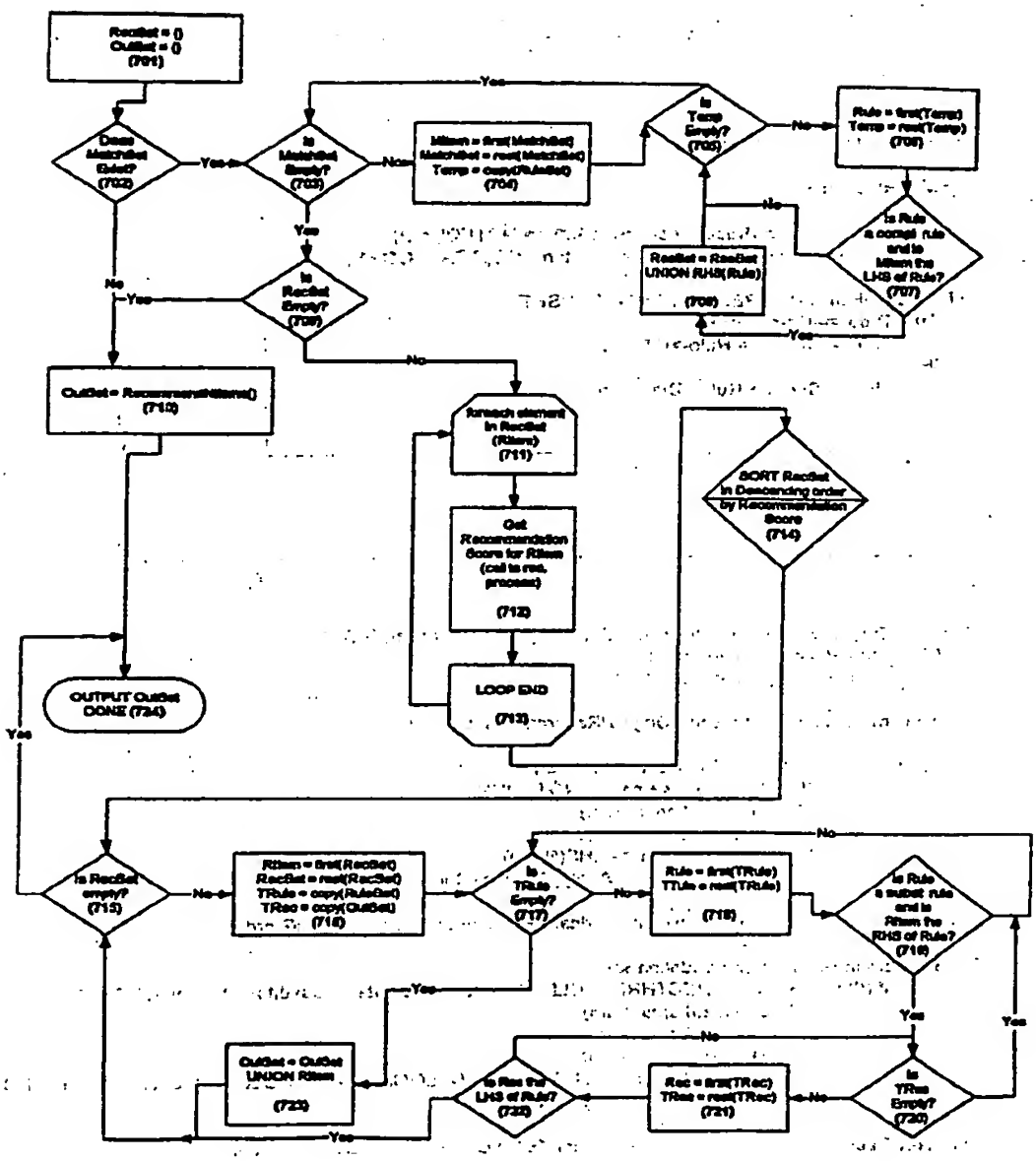

RecSet = {}
OutSet = {}
(701)
Done MatchSet Exist? (702)
Yes
No
Is MatchSet Empty? (703)
Yes
No
RItem = first(MatchSet)
MatchSet = rest(MatchSet)
Temp = copy(RuleSet)
(704)
Is Temp Empty? (705)
Yes
No
Rule = first(Temp)
Temp = rest(Temp)
(706)
Is Rule a concept rule and is RItem the LHS of Rule? (707)
Yes
No
RecSet = RecSet UNION RHS(Rule)
(708)
Is RecSet Empty? (709)
Yes
No
OutSet = RecommendedItems()
(710)
foreach element in RecSet (RItem) (711)
Get Recommendation Score for RItem (call to rec. process)
(712)
LOOP END
(713)
SORT RecSet in Descending order by Recommendation Score
(714)
OUTPUT OutSet DONE (734)
Is RecSet empty? (715)
Yes
No
RItem = first(RecSet)
RecSet = rest(RecSet)
TRule = copy(RuleSet)
TRec = copy(OutSet)
(716)
Is TRule Empty? (717)
Yes
No
Rule = first(TRule)
TRule = rest(TRule)
(718)
Is Rule a subset rule and is RItem the RHS of Rule? (719)
Yes
No
Is TRec Empty? (720)
Yes
No
Rec = first(TRec)
TRec = rest(TRec)
(721)
Is Rec the LHS of Rule? (722)
Yes
No
OutSet = OutSet UNION RItem
(723)

FIG. 7B

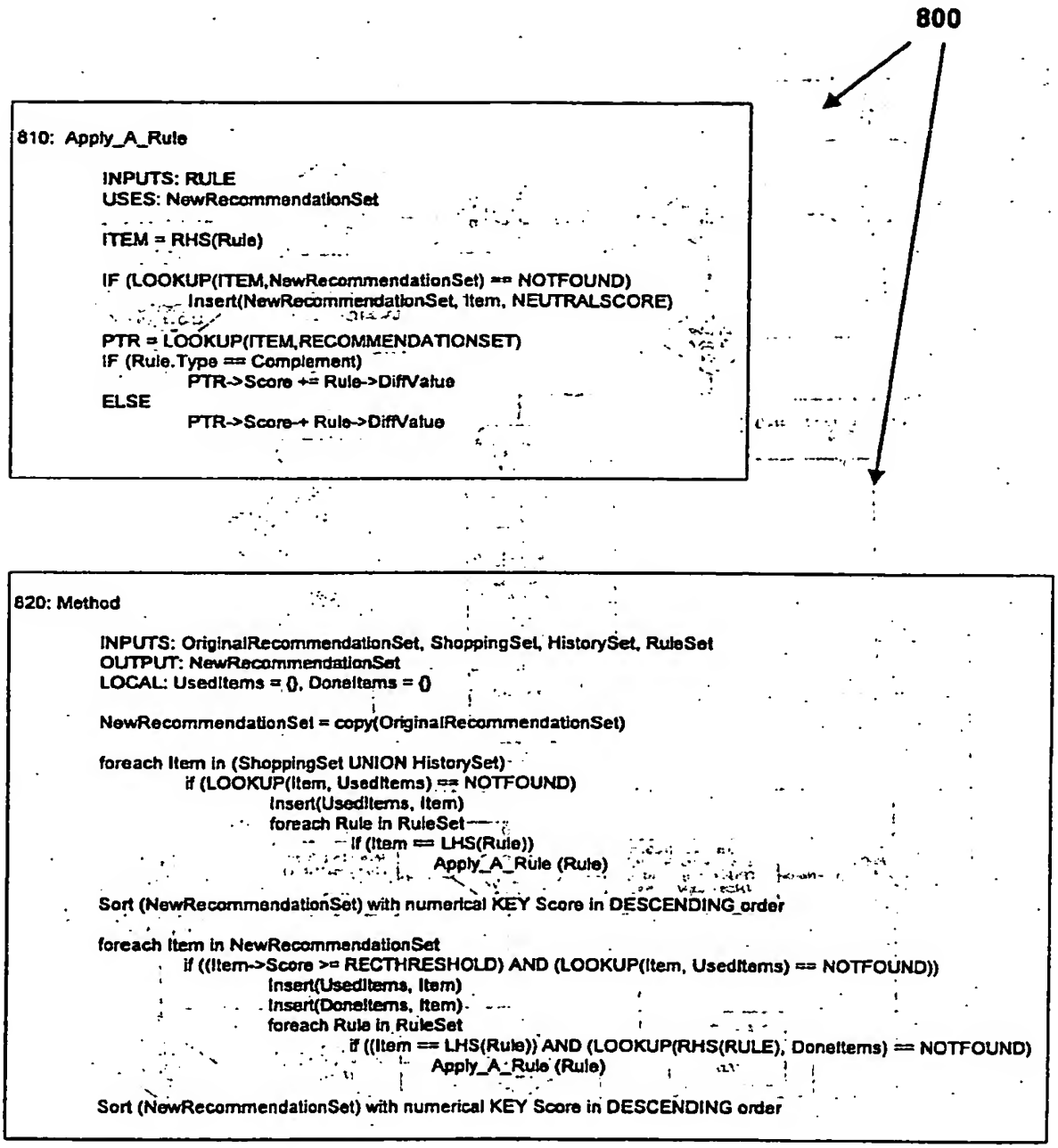
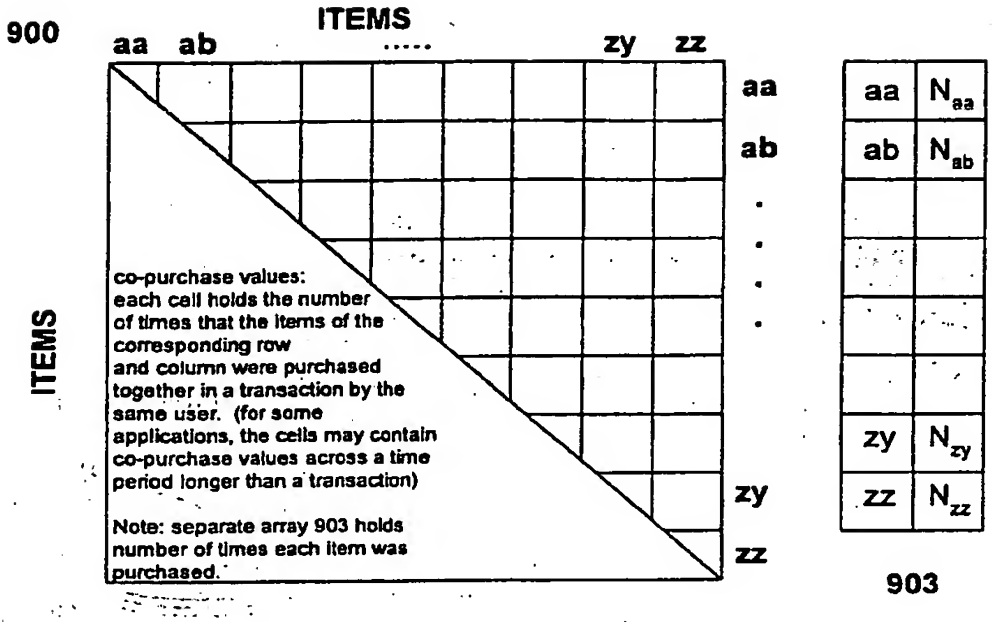

800
810: Apply_A_Rule
INPUTS: RULE
USES: NewRecommendationSet
ITEM = RHS(Rule)
IF (LOOKUP(ITEM,NewRecommendationSet) == NOTFOUND)
Insert(NewRecommendationSet, Item, NEUTRALSCORE)
PTR = LOOKUP(ITEM,RECOMMENDATIONSET)
IF (Rule.Type == Complement)
PTR->Score += Rule->DiffValue
ELSE
PTR->Score-+ Rule->DiffValue
820: Method
INPUTS: OriginalRecommendationSet, ShoppingSet, HistorySet, RuleSet
OUTPUT: NewRecommendationSet
LOCAL: UsedItems = {}, DoneItems = {}
NewRecommendationSet = copy(OriginalRecommendationSet)
foreach Item in (ShoppingSet UNION HistorySet)
if (LOOKUP(Item, UsedItems) == NOTFOUND)
Insert(UsedItems, Item)
foreach Rule in RuleSet
if (Item == LHS(Rule))
Apply_A_Rule (Rule)
Sort (NewRecommendationSet) with numerical KEY Score in DESCENDING order
foreach Item in NewRecommendationSet
if ((Item->Score >= RECTHRESHOLD) AND (LOOKUP(Item, UsedItems) == NOTFOUND))
Insert(UsedItems, Item)
Insert(DoneItems, Item)
foreach Rule in RuleSet
if ((Item == LHS(Rule)) AND (LOOKUP(RHS(RULE), DoneItems) == NOTFOUND)
Apply_A_Rule (Rule)
Sort (NewRecommendationSet) with numerical KEY Score in DESCENDING order

FIG. 8




```
901: Method for inferring substitute rules

Foreach row from 1 to NUMITEMS
   Foreach column from row+1 to NUMITEMS
      if ((value(row, column] / TotalPurchases(row) < SubThreshold)
         AND
         (TotalPurchases(row) > SubSignifThreshold))
      then
        Add_Rule (row -> NOT column);

      if ((value[row,column] / TotalPurchases(column) < SubThreshold)
         AND
         (TotalPurchases(column) > SubSignifThreshold))
      then
        Add_Rule (column -> NOT row);
```

```
902: Method for inferring complement rules

Foreach row from 1 to NUMITEMS
   Foreach column from row+1 to NUMITEMS
      if ((value(row, column] / TotalPurchases(row) > CompThreshold)
         AND
         (TotalPurchases(row) > CompSignifThreshold))
      then
        Add_Rule (row -> column);

      if ((value[row,column] / TotalPurchases(column) > CompThreshold)
         AND
         (TotalPurchases(column) > CompSignifThreshold))
      then
        Add_Rule (column -> row);
```

FIG. 9

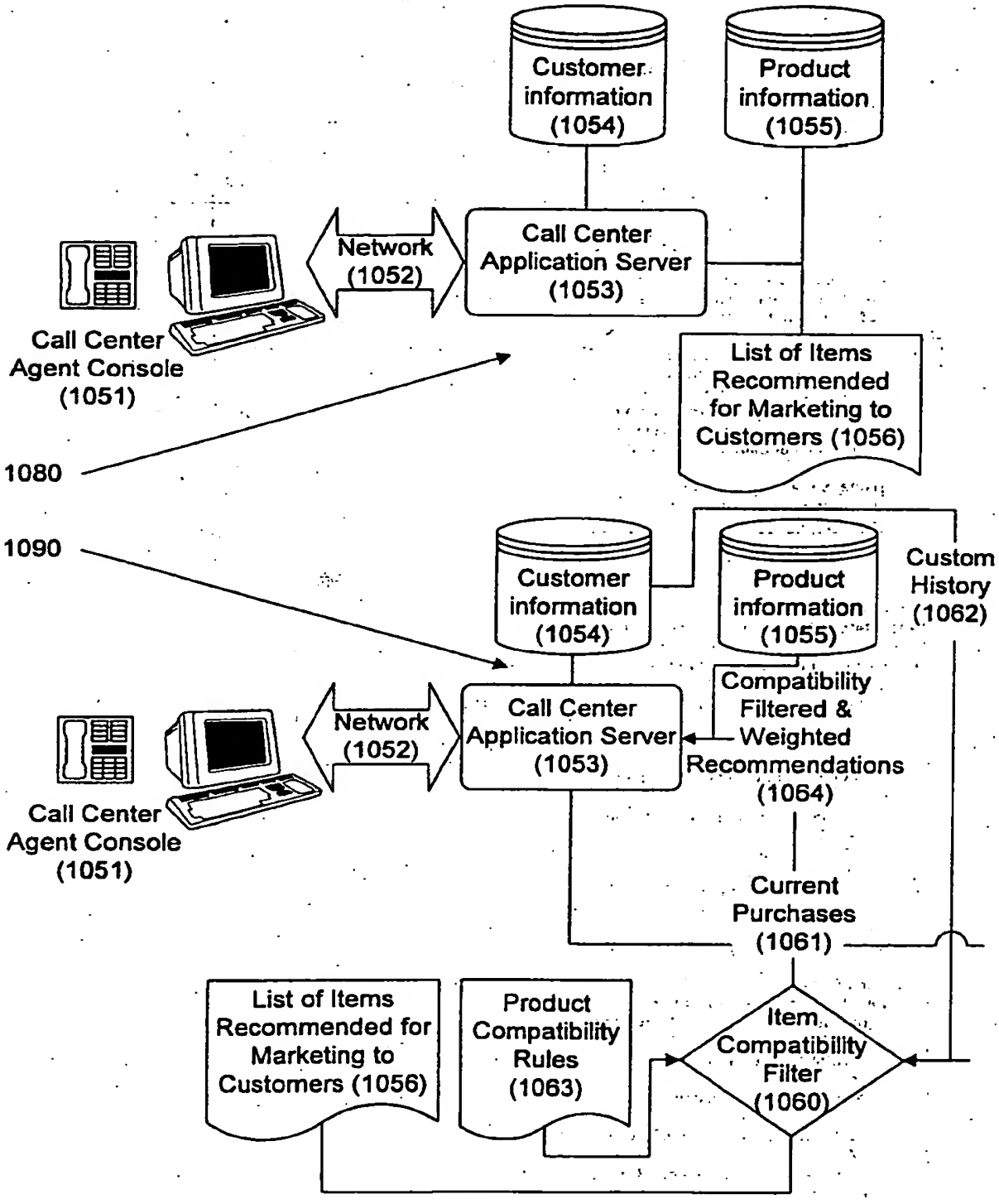

Customer information (1054)
Product information (1055)
Network (1052)
Call Center Application Server (1053)
Call Center Agent Console (1051)
List of Items Recommended for Marketing to Customers (1056)
1080
1090
Customer information (1054)
Product information (1055)
Custom History (1062)
Network (1052)
Call Center Application Server (1053)
Compatibility Filtered & Weighted Recommendations (1064)
Call Center Agent Console (1051)
Current Purchases (1061)
List of Items Recommended for Marketing to Customers (1056)
Product Compatibility Rules (1063)
Item Compatibility Filter (1060)

FIG. 10

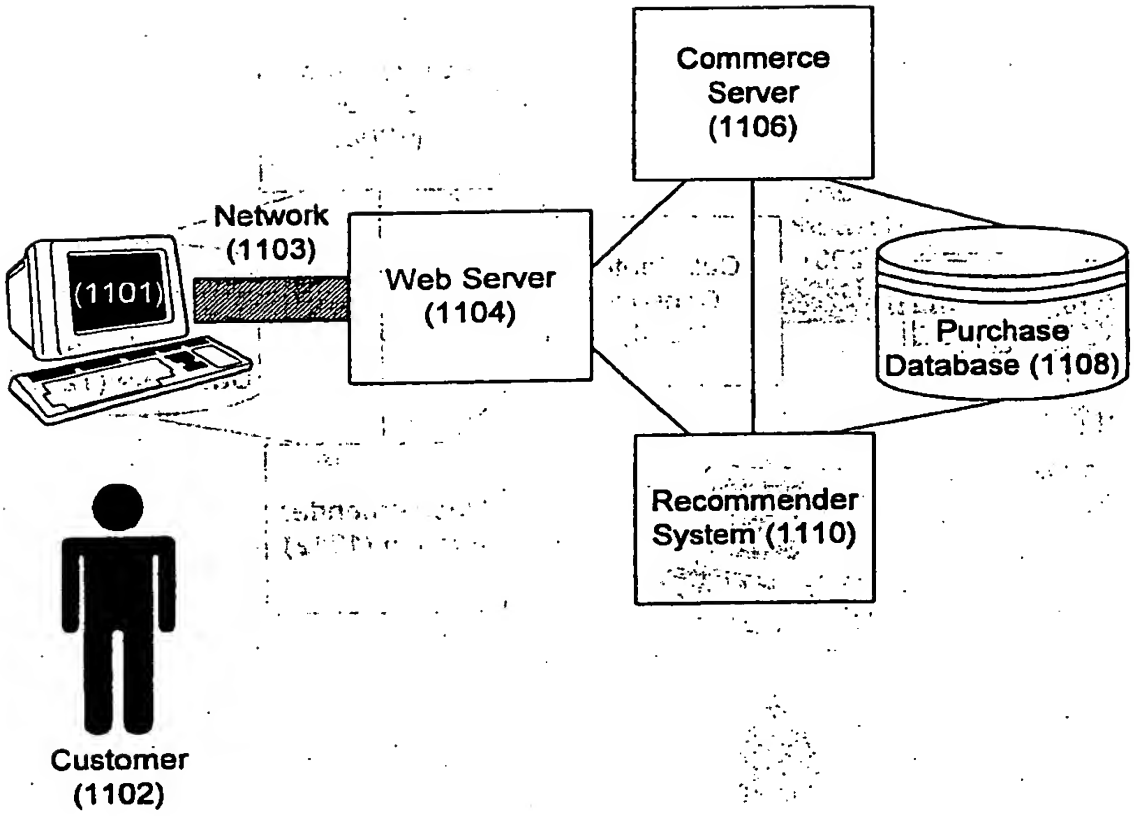

Commerce Server (1106)
Network (1103)
(1101)
Web Server (1104)
Purchase Database (1108)
Recommender System (1110)
Customer (1102)

FIG. 11

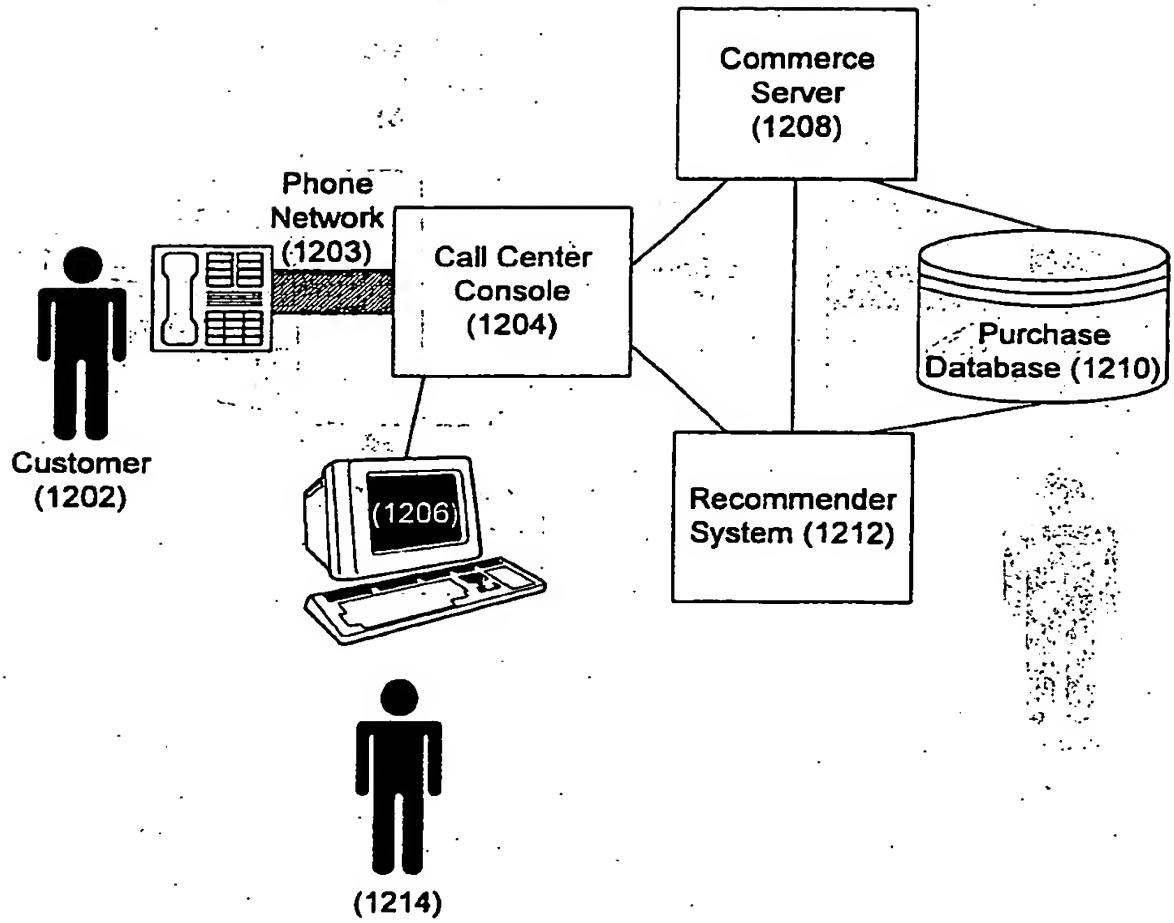

Commerce Server (1208)
Phone Network (1203)
Call Center Console (1204)
Purchase Database (1210)
Customer (1202)
(1206)
Recommender System (1212)
(1214)

FIG. 12

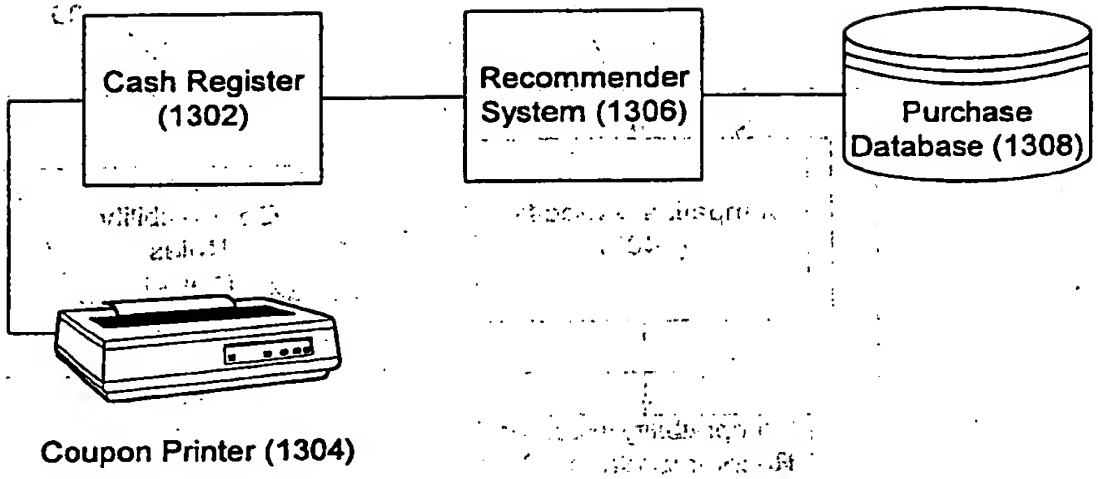

Cash Register (1302)
Recommender System (1306)
Purchase Database (1308)
Coupon Printer (1304)

FIG. 13

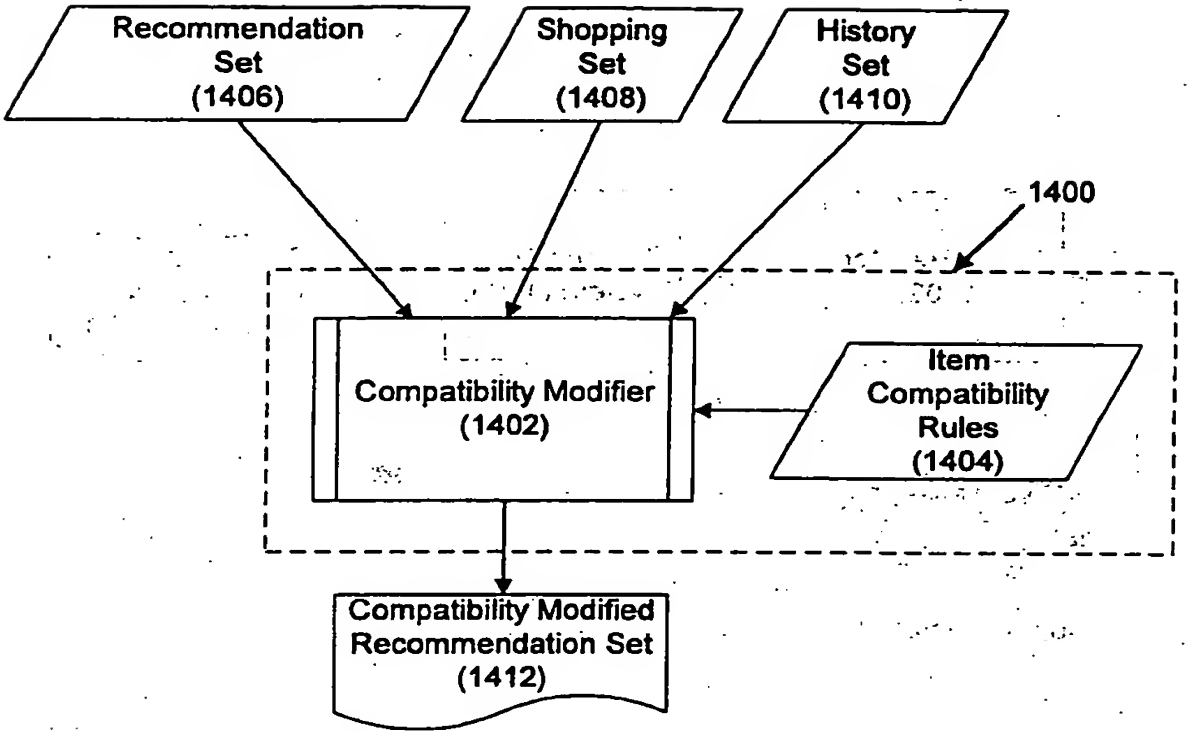

Recommendation Set (1406)
Shopping Set (1408)
History Set (1410)
1400
Compatibility Modifier (1402)
Item Compatibility Rules (1404)
Compatibility Modified Recommendation Set (1412)

FIG. 14A

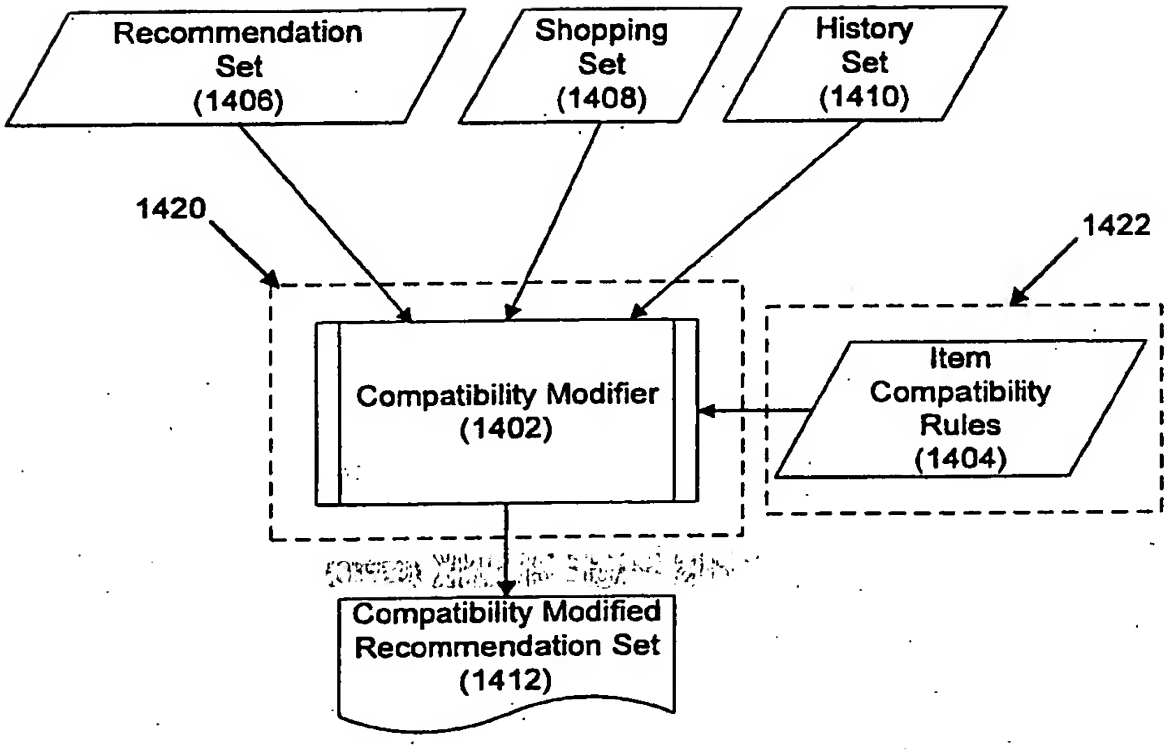

Recommendation Set (1406)
Shopping Set (1408)
History Set (1410)
1420
1422
Compatibility Modifier (1402)
Item Compatibility Rules (1404)
Compatibility Modified Recommendation Set (1412)

FIG. 14B